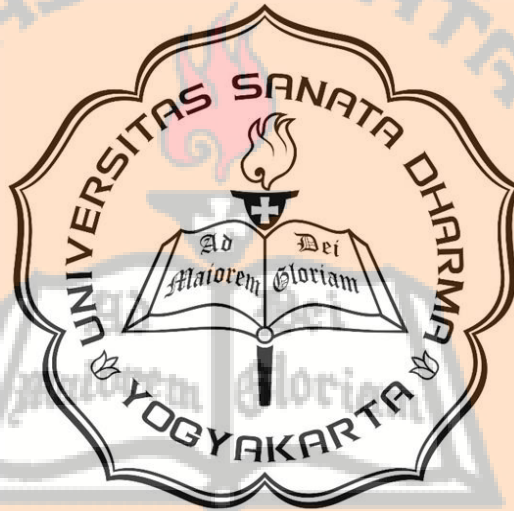


**PENERAPAN METODE *RANDOM FOREST*
UNTUK PREDIKSI *WIN RATIO*
PEMAIN *PLAYER UNKNOWN BATTLEGROUN*
SKRIPSI**

Diajukan untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
Program Studi Teknik Informatika



Oleh:

Reinardus Aji Haristu

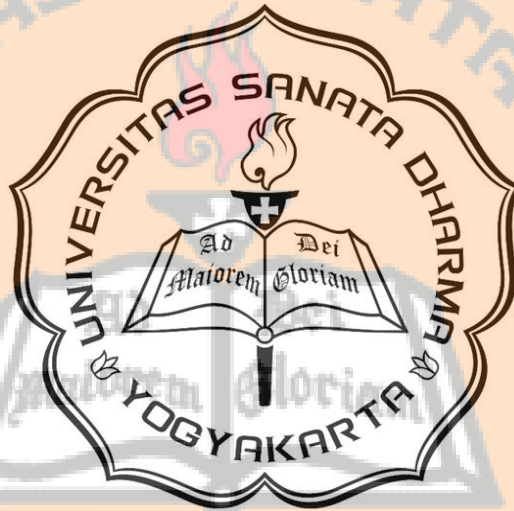
155314090

**PROGRAM STUDI TEKNIK INFORMATIKA
JURUSAN TEKNIK INFORMATIKA
FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS SANATA DHARMA
YOGYAKARTA**

2019

**PENERAPAN METODE *RANDOM FOREST*
UNTUK PREDIKSI *WIN RATIO*
PEMAIN *PLAYER UNKNOWN BATTLEGROUN*
SKRIPSI**

Diajukan untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
Program Studi Teknik Informatika



Oleh:

Reinardus Aji Haristu

155314090

**PROGRAM STUDI TEKNIK INFORMATIKA
JURUSAN TEKNIK INFORMATIKA
FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS SANATA DHARMA
YOGYAKARTA**

2019

**IMPLEMENTATION OF RANDOM FOREST ALGORITHM
TO PREDICT WIN RATIO OF
“PLAYER UNKNOWN BATTLEGROUNDS” PLAYER
FINAL PROJECT**

Present as Partial Fullfillment of the Requirements
to Obtain the *Sarjana Komputer* Degree
in Informatics Engineering Study Program



By:

Reinardus Aji Haristu

155314090

**INFORMATICS ENGINEERING STUDY PROGRAM
DEPARTMENT OF INFORMATICS ENGINEERING
FACULTY OF SCIENCE AND TECHNOLOGY
SANATA DHARMA UNIVERSITY
YOGYAKARTA**

2019

HALAMAN PERSETUJUAN

SKRIPSI

PENERAPAN METODE *RANDOM FOREST*
UNTUK PREDIKSI *WIN RATIO*
PEMAIN *PLAYER UNKNOWN BATTLEGROUNDS*

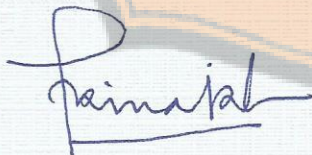
Oleh:

Reinardus Aji Haristu

155314090

Telah Disetujui Oleh:

Dosen Pembimbing,



P.H. Prima Rosa, S.Si., M.Sc.

Tanggal, 25 Juli 2019

HALAMAN PENGESAHAN
SKRIPSI
PENERAPAN METODE *RANDOM FOREST*
UNTUK PREDIKSI *WIN RATIO*
PEMAIN *PLAYER UNKNOWN BATTLEGROUN*

Dipersiapkan dan ditulis oleh:

REINARDUS AJI HARISTU

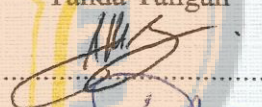
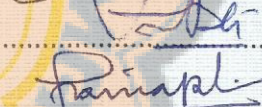
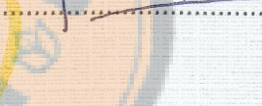
NIM : 155314090

Telah dipertahankan di depan Panitia Penguji

Pada tanggal 10 Juli 2019..

Dan dinyatakan memenuhi syarat

Susunan Panitia Penguji

	Nama Lengkap	Tanda Tangan
Ketua	Robertus Adi Nugroho, S.T., M.Eng.	
Sekretaris	Puspaningtyas Sanjoyo Adi, S.T., M.T.	
Anggota	P.H. Prima Rosa, S.Si., M.Sc.	


Yogyakarta, 26 Juli 2019

Fakultas Sains dan Teknologi

Universitas Sanata Dharma

Dekan




Sudi Mungkasi, S.Si, M.Math.Sc., Ph.D

HALAMAN PERSEMBAHAN

*“Hidup itu seperti pertunjukan wayang, dimana
kamu menjadi dalang atas naskah semesta yang
dituliskan oleh Tuhan mu”*

(Sujiwo Tedjo)

Karya ini kupersembahkan kepada:

Tuhan Yesus Kristus

Bunda Maria

Keluarga

Sahabat

PERNYATAAN KEASLIAN KARYA

Saya menyatakan dengan sesungguhnya bahwa skripsi yang saya tulis ini tidak memuat karya atau bagian karya orang lain, kecuali yang telah saya sebutkan dalam kutipan daftar pustaka, sebagaimana layaknya karya ilmiah.

Yogyakarta, 26 Juli 2019

Penulis,



Reinardus Aji Haristu

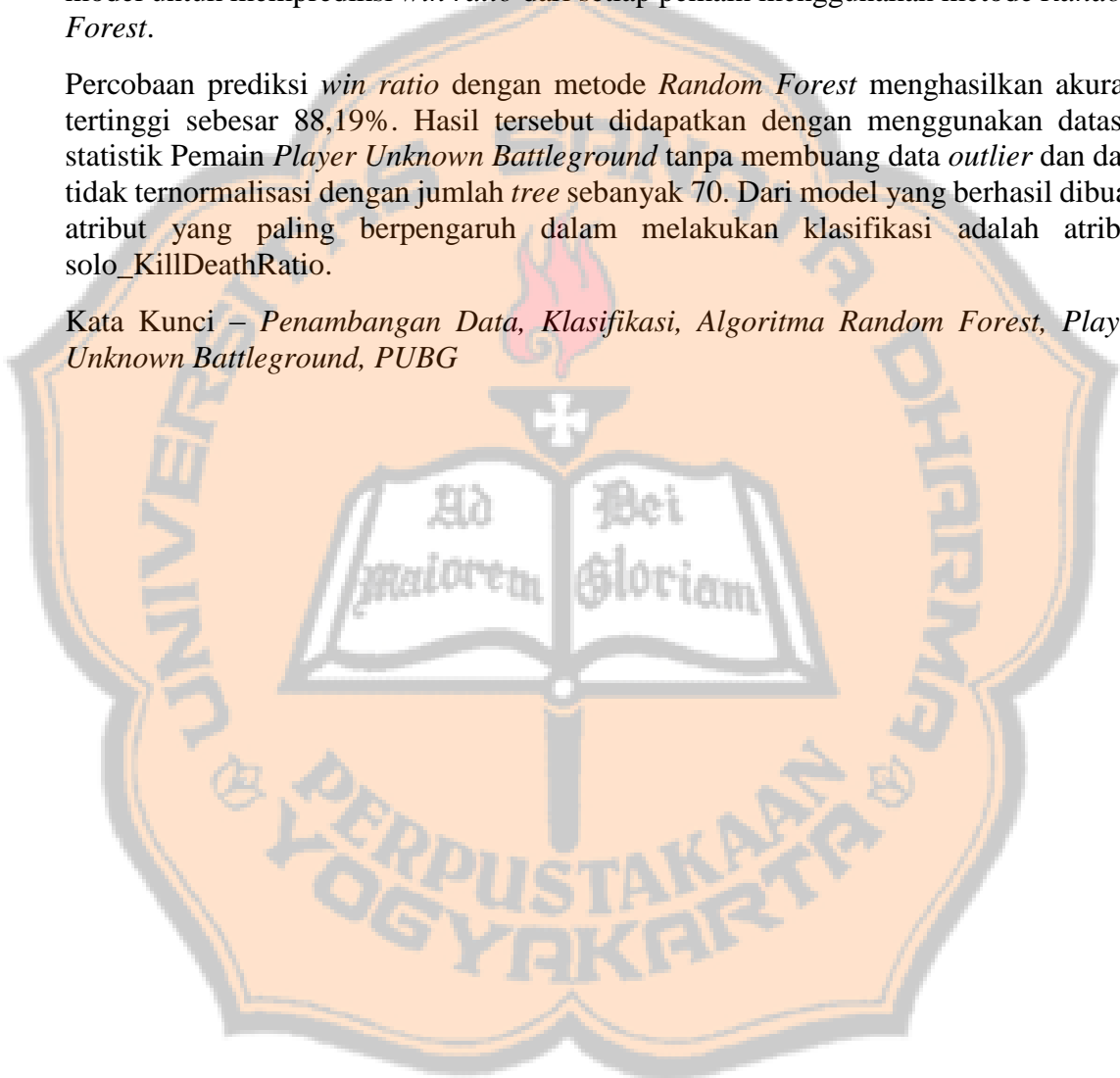


ABSTRAK

Game online yang sedang naik daun sebagai salah satu *e-sport* adalah *Player Unknown Battleground*. Untuk memenangkan *game* tersebut dibutuhkan strategi yang tepat. Dalam tugas akhir ini diuraikan penelitian tentang strategi bermain yang efektif pada *game Player Unknown Battleground* dengan melakukan penambangan data terhadap data statistik pemain yang diambil dari website kaggle. Dari data tersebut akan dibuat model untuk memprediksi *win ratio* dari setiap pemain menggunakan metode *Random Forest*.

Percobaan prediksi *win ratio* dengan metode *Random Forest* menghasilkan akurasi tertinggi sebesar 88,19%. Hasil tersebut didapatkan dengan menggunakan dataset statistik Pemain *Player Unknown Battleground* tanpa membuang data *outlier* dan data tidak ternormalisasi dengan jumlah *tree* sebanyak 70. Dari model yang berhasil dibuat, atribut yang paling berpengaruh dalam melakukan klasifikasi adalah atribut *solo_KillDeathRatio*.

Kata Kunci – *Penambangan Data, Klasifikasi, Algoritma Random Forest, Player Unknown Battleground, PUBG*

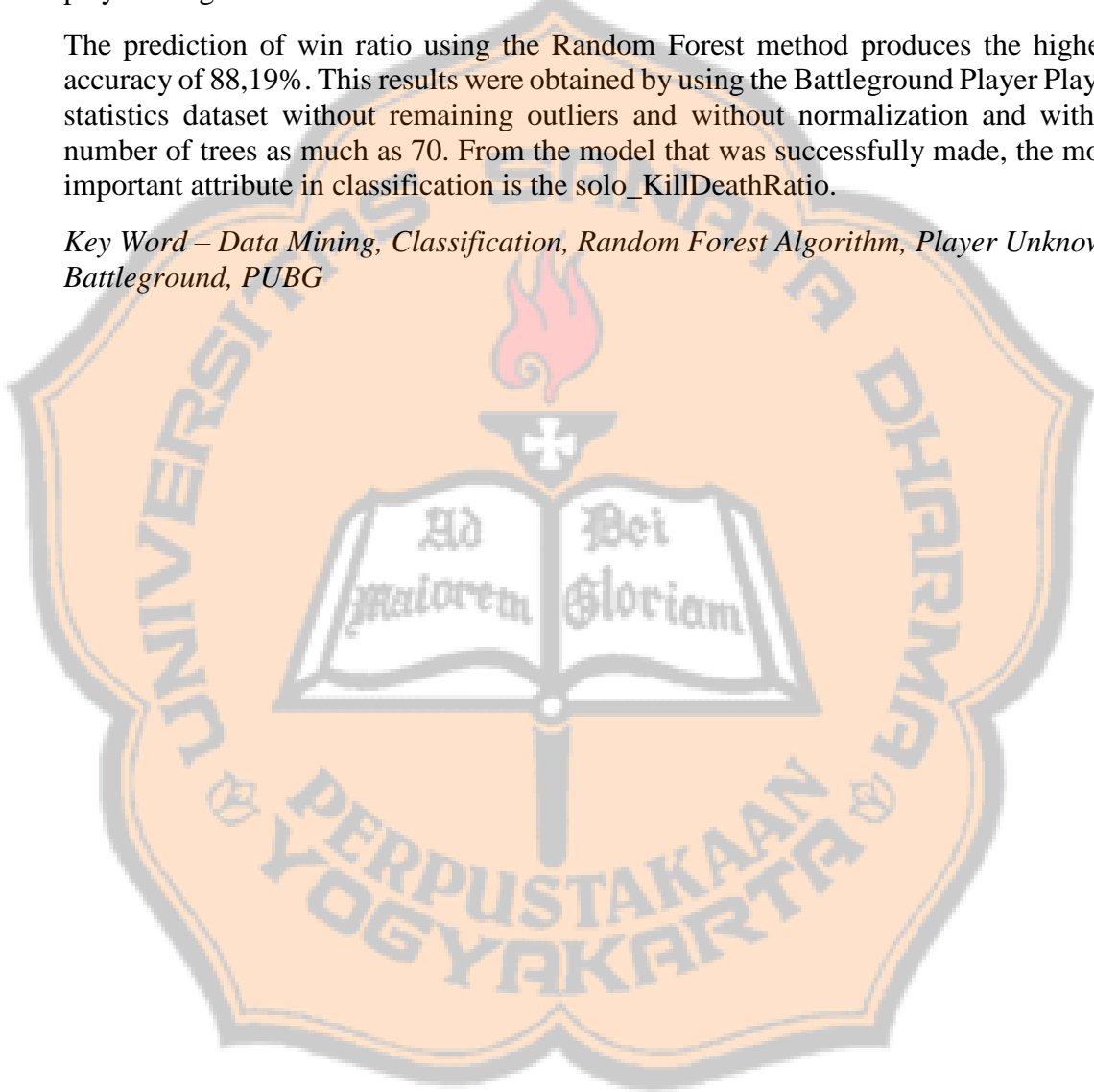


ABSTRACT

PUBG online game is one of the rising e-sport. To win the game, the right strategy is needed. In this final assignment, a study of effective playing strategies on Unknown Battleground game is carried out by mining player statistical data taken from the website kaggle. From the data, a model was be created to predict the win ratio of each player using the Random Forest method.

The prediction of win ratio using the Random Forest method produces the highest accuracy of 88,19%. This results were obtained by using the Battleground Player Player statistics dataset without remaining outliers and without normalization and with a number of trees as much as 70. From the model that was successfully made, the most important attribute in classification is the solo_KillDeathRatio.

Key Word – Data Mining, Classification, Random Forest Algorithm, Player Unknown Battleground, PUBG



**LEMBAR PERNYATAAN PERSETUJUAN PUBLIKASI KARYA ILMIAH
UNTUK KEPERLUAN KEPENTINGAN AKADEMIS**

Yang bertanda tangan di bawah ini, saya mahasiswa Universitas Sanata Dharma :

Nama : Reinardus Aji Haristu

NIM : 155314090

Demi pengembangan ilmu pengetahuan, saya memberikan kepada Perpustakaan Universitas Sanata Dharma karya ilmiah saya yang berjudul:

**PENERAPAN METODE *RANDOM FOREST*
UNTUK PREDIKSI *WIN RATIO*
PEMAIN *PLAYER UNKNOWN BATTLEGROUN***

Beserta perangkat yang diperlukan (bila ada). Dengan demikian saya memberikan kepada Perpustakaan Universitas Sanata Dharma hak untuk menyimpan, mengalihkan dalam bentuk media lain, mengelola di internet atau media lain untuk kepentingan akademis tanpa perlu meminta ijin dari saya maupun memberikan royalti kepada saya selama tetap mencantumkan nama saya sebagai penulis.

Demikian pernyataan ini saya buat dengan sebenarnya

Dibuat di Yogyakarta

Pada tanggal26 Juli.....2019

Yang menyatakan,



Reinardus Aji Haristu

KATA PENGANTAR


Puji Syukur penulis haturkan kepada Tuhan Yesus Kristus atas rahmat-Nya, sehingga penulis dapat menyelesaikan Tugas Akhir yang berjudul “PENERAPAN METODE RANDOM FOREST UNTUK PREDIKSI *WIN RATIO* PEMAIN *PLAYER UNKNOWN BATTLEGROUNDS*”.

Dalam penyelesaian Tugas Akhir ini, penulis mendapatkan banyak dukungan, doa dan motivasi dari berbagai pihak, untuk itu penulis mengucapkan terimakasih kepada:

1. Tuhan Yesus Kristus yang selalu memberikan kekuatan dan kehidupan yang baru setiap harinya
2. Ibu P.H. Prima Rosa, S.Si., M.Sc. selaku Dosen pembimbing yang selalu sabar dalam membimbing dan mengarahkan selama penyusunan tugas akhir.
3. Bapak Drs. Johanes Eka Priyatma, M.Sc., Ph.D. selaku Rektor Universitas Sanata Dharma Yogyakarta.
4. Bapak Sudi Mungkasi, S.Si, M.Math.Sc., Ph.D selaku Dekan Fakultas Sains dan Teknologi Universitas Sanata Dharma.
5. Ibu Dr. Anastasia Rita Widiarti selaku Ketua Program Studi Teknik Informatika Fakultas Sains dan Teknologi Universitas Sanata Dharma.
6. Orang tua, Vincentius Haryono dan Cicilia Suharni yang selalu ada mendukung penulis setiap saat.
7. Maria Damayanti yang selalu ada untuk menemani penulis dalam menyusun tugas akhir ini dan memberikan banyak saran kepada penulis.
8. Seluruh teman-teman TI 2015 tanpa terkecuali yang sudah membantu dan mendukung dalam pembuatan tugas akhir ini.
9. Seluruh pihak yang sudah mendukung secara langsung maupun tidak langsung, maaf apabila tidak dapat menyebutkan satu per satu.

Penulis berharap penulisan ini menjadi pengetahuan baru yang berguna dan bermanfaat bagi para pembaca. Penulis menyadari bahwa penulisan tugas akhir ini masih sangat banyak kekurangan. Untuk itu, penulis sangat berharap saran dan kritik untuk perbaikan di masa yang akan datang.

Yogyakarta, 26 Juli 2019
Penulis,



Reinardus Aji Haristu



DAFTAR ISI

HALAMAN PERSETUJUAN.....	iii
HALAMAN PENGESAHAN.....	iv
HALAMAN PERSEMBAHAN	v
PERNYATAAN KEASLIAN KARYA	vi
ABSTRAK.....	vii
ABSTRACT.....	viii
LEMBAR PERNYATAAN PERSETUJUAN PUBLIKASI KARYA ILMIAH.....	ix
KATA PENGANTAR	x
DAFTAR ISI.....	xii
DAFTAR GAMBAR	xvi
DAFTAR TABEL.....	xviii
BAB I PENDAHULUAN.....	1
1.1. Latar Belakang	1
1.2. Rumusan Masalah	4
1.3. Tujuan.....	4
1.4. Batasan Masalah.....	4
1.5. Sistematika Penulisan.....	5
BAB II TINJAUAN PUSTAKA.....	6
2.1. Penambangan Data (<i>Data Mining</i>).....	6
2.2. <i>Metode Random Forest</i>	8
2.3. <i>Classification And Regression Tree (CART)</i>	12
2.4. Evaluasi Kerja	14

2.4.1.	<i>K-Fold Cross Validation</i>	14
2.4.2.	<i>Confusion Matrix</i>	15
BAB III METODOLOGI PENELITIAN		17
3.1.	Bahan Penelitian/Data	17
3.2.	Peralatan Penelitian	21
3.3.	Tahap-Tahap Penelitian.....	21
3.1.1.	Pengumpulan Data	22
3.1.2.	Penelitian Pustaka	22
3.1.3.	<i>Knowledge Discovery in Database (KDD)</i>	22
3.1.4.	Pengembangan Perangkat Lunak	23
3.1.5.	Analisis dan Pembuatan Laporan.....	25
BAB IV PEMROSESAN AWAL DAN PERANCANGAN SISTEM.....		26
4.1.	Pemrosesan Awal	26
4.1.1.	Seleksi data (<i>Data Selection</i>)	26
4.1.2.	Transformasi data (<i>Data Transformation</i>).....	29
4.2.	Perancangan Perangkat Lunak	31
4.2.1.	Analisis Kebutuhan Perangkat Lunak.....	31
4.2.2.	Diagram <i>Use Case</i>	33
4.2.3.	Narasi <i>Use Case</i>	33
4.2.4.	Diagram Kelas Analisis.....	33
4.2.5.	Diagram Kelas UML.....	35
4.2.6.	Perancangan Struktur Data.....	36
4.2.7.	Algoritma per Metode	37
4.2.8.	Perancangan Antarmuka	43

BAB V IMPLEMENTASI DAN ANALISIS HASIL	45
5.1. Implementasi Perangkat Lunak	45
5.1.1. Implementasi Kelas <i>Model</i>	45
5.1.1. Implementasi Kelas <i>Control</i>	45
5.1.2. Implementasi Kelas <i>View</i>	46
5.2. Analisis Hasil	46
5.2.1. Pengujian Perangkat Lunak.....	46
5.2.1.1. Prosedur Pengujian dan Kasus Uji.....	46
5.2.1.2. Evaluasi Pengujian.....	46
5.2.2. Pengujian Perbandingan Hasil Hitung Manual dengan Hasil Perangkat Lunak 47	
5.2.2.1. Penghitungan Perangkat Lunak	47
5.2.2.2. Penghitungan Manual	51
5.2.2.3. Evaluasi Pengujian.....	51
5.2.3. Pengujian Perangkat Lunak Dengan Menggunakan <i>Dataset</i>	54
5.2.3.1. Pengujian <i>Dataset</i> statistik Pemain <i>Player Unknown Battleground</i> dengan <i>outlier</i> dan data tidak ternormalisasi.	54
5.2.3.2. Pengujian <i>Dataset</i> statistik Pemain <i>Player Unknown Battleground</i> dengan <i>outlier</i> dan data ternormalisasi.	55
5.2.3.3. Pengujian <i>Dataset</i> statistik Pemain <i>Player Unknown Battleground</i> tanpa <i>outlier</i> dan data tidak ternormalisasi.	56
5.2.3.4. Pengujian <i>Dataset</i> statistik Pemain <i>Player Unknown Battleground</i> tanpa <i>outlier</i> dan data ternormalisasi.	57
5.2.3.5. Evaluasi Hasil Pengujian	57
BAB VI PENUTUP	62

6.1. Kesimpulan.....	62
6.2. Saran	62
DAFTAR PUSTAKA	63
LAMPIRAN I: NARASI <i>USE CASE</i>	66
LAMPIRAN II: PROSEDUR PENGUJIAN DAN KASUS UJI.....	72
LAMPIRAN III: HITUNG MANUAL PENAMBANGAN DATA.....	77



DAFTAR GAMBAR

Gambar 2. 1 <i>Dataset</i>	10
Gambar 2. 2 <i>Random Dataset</i> untuk <i>tree</i> 1	10
Gambar 2. 3 <i>Random Dataset</i> untuk <i>tree</i> 2.....	10
Gambar 2. 4 <i>Random Dataset</i> untuk <i>tree</i> 3.....	10
Gambar 3. 1 Skema penelitian	22
Gambar 4. 1 Hasil Klustering <i>Tools</i> Weka Untuk Atribut solo_WinRatio.....	30
Gambar 4. 2 Hasil Klustering <i>Tools</i> Weka Untuk Atribut solo_RoundsPlayed	30
Gambar 4. 3 Hasil Klustering <i>Tools</i> Weka Untuk Atribut solo_WinRatio Setelah Menghilangkan Kluster 0 atribut solo_RoundsPlayed.....	30
Gambar 4. 4 Diagram <i>Flowchart</i>	32
Gambar 4. 5 Contoh Hasil Visualisasi <i>Tree</i>	32
Gambar 4. 6 Diagram <i>Use Case</i>	33
Gambar 4. 7 Diagram Model Kelas Analisis	34
Gambar 4. 8 Diagram Kelas UML.....	35
Gambar 4. 9 Desain Halaman Awal.....	43
Gambar 4. 10 Desain Halaman Preprocessing.....	43
Gambar 4. 11 Desain Halaman Initialization.....	44
Gambar 4. 12 Desain Halaman Single Data Test.....	44
Gambar 5. 1 Preprocessing Penghitungan Perangkat Lunak	47
Gambar 5. 2 Hasil Akurasi Penghitungan Sistem.....	48
Gambar 5. 3 Hasil Pembentukan <i>Tree</i> 1 Pada Model 1 Oleh Sistem	48

Gambar 5. 4 Hasil Pembentukan <i>Tree</i> 2 Pada Model 1 Oleh Sistem	49
Gambar 5. 5 Hasil Pembentukan <i>Tree</i> 1 Pada Model 2 Oleh Sistem	49
Gambar 5. 6 Hasil Pembentukan <i>Tree</i> 2 Pada Model 2 Oleh Sistem	49
Gambar 5. 7 Hasil Sampling Dataset Untuk <i>Tree</i> 1 Pada Model 1 Oleh Sistem	49
Gambar 5. 8 Hasil Sampling Dataset Untuk <i>Tree</i> 2 Pada Model 1 Oleh Sistem	50
Gambar 5. 9 Hasil Sampling Dataset Untuk <i>Tree</i> 1 Pada Model 2 Oleh Sistem	50
Gambar 5. 10 Hasil Sampling Dataset Untuk <i>Tree</i> 2 Pada Model 2 Oleh Sistem	50
Gambar 5. 11 Hasil pengujian dataset statistik pemain <i>Player Unknown Battleground</i>	58
Gambar 5. 12 Hasil pengujian dataset statistik pemain <i>Player Unknown Battleground</i> dengan outlier dan data ternormalisasi	58
Gambar 5. 13 Hasil pengujian dataset statistik pemain <i>Player Unknown Battleground</i> tanpa outlier dan data tidak ternormalisasi	59
Gambar 5. 14 Hasil pengujian dataset statistik pemain <i>Player Unknown Battleground</i> tanpa outlier dan data ternormalisasi	59
Gambar 5. 15 Korelasi atribut solo_ KillDeathRatio dengan <i>win ratio</i>	61
Gambar 6. 1 Hasil Akhir <i>Tree</i> 1 Pada Model 1 Penghitungan Manual	119
Gambar 6. 2 Hasil Akhir <i>Tree</i> 2 Pada Model 1 Penghitungan Manual	119
Gambar 6. 3 Hasil Akhir <i>Tree</i> 1 Pada Model 2 Penghitungan Manual	121
Gambar 6. 4 Hasil Akhir <i>Tree</i> 2 Pada Model 2 Penghitungan Manual	121

DAFTAR TABEL

Tabel 2. 1 Matrik Klasifikasi untuk Model dua kelas	16
Tabel 3. 1 Penjelasan Atribut dan Contoh Data Statistik Pemain PUBG	17
Tabel 4. 1 Daftar atribut yang dihapus.....	26
Tabel 4. 2 Tabel Atribut Data Statistik Pemain <i>Player Unknown Battleground</i> Yang Akan Digunakan.....	28
Tabel 5. 1 Implementasi kelas <i>model</i>	45
Tabel 5. 2 Implementasi kelas <i>control</i>	45
Tabel 5. 3 Implementasi kelas <i>view</i>	46
Tabel 5. 4 Perbandingan <i>Tree</i> Hasil Sistem dan Perhitungan Manual.....	52
Tabel 5. 5 Pengujian dataset statistik Pemain <i>Player Unknown Battleground</i>	55
Tabel 5. 6 Pengujian <i>dataset</i> statistik Pemain <i>Player Unknown Battleground</i> dengan outlier dan data ternormalisasi	56
Tabel 5. 7 Pengujian <i>dataset</i> statistik Pemain <i>Player Unknown Battleground</i> tanpa outlier dan data tidak ternormalisasi	56
Tabel 5. 8 Pengujian <i>dataset</i> statistik Pemain <i>Player Unknown Battleground</i> dengan outlier dan data ternormalisasi	57
Tabel 6. 1 Narasi <i>Use Case</i> Input <i>Dataset</i>	66
Tabel 6. 2 Narasi <i>Use Case</i> Seleksi Atribut	67
Tabel 6. 3 Narasi <i>Use Case</i> Inisialisasi Model.....	68
Tabel 6. 4 Narasi <i>Use Case</i> Proses Klasifikasi Dengan Metode Random Forest	69
Tabel 6. 5 Narasi <i>Use Case</i> Lihat Model <i>Tree</i>	70

Tabel 6. 6 Narasi Use Case Uji Data Tunggal	71
Tabel 6. 7 Prosedur Pengujian dan Kasus Uji.....	72
Tabel 6. 8 Dataset yang digunakan untuk membangun model	77
Tabel 6. 9 Pembagian Data Training dan Data Testing Setiap Model.....	78
Tabel 6. 10 Random dataset untuk membangun <i>tree</i> 1 pada model 1	78
Tabel 6. 11 Random dataset untuk membangun <i>tree</i> 2 pada model 1	89
Tabel 6. 12 Random dataset untuk membangun <i>tree</i> 1 pada model 2	97
Tabel 6. 13 Random dataset untuk membangun <i>tree</i> 2 pada model 2	108
Tabel 6. 14 Hasil Klasifikasi Data Testing Model 1	120
Tabel 6. 15 Evaluasi Hasil Model 1	120
Tabel 6. 16 Hasil Klasifikasi Data Testing Model 2.....	122
Tabel 6. 17 Evaluasi Hasil Model 1	122

BAB I PENDAHULUAN

1.1. Latar Belakang

Perkembangan teknologi di zaman sekarang ini sudah maju sangat pesat. Berbagai inovasi dan terobosan dibidang teknologi diciptakan. Komputer tidak lagi digunakan untuk menghitung (*compute*). Dalam dunia multimedia, komputer diperlukan untuk pembuatan film dengan animasi yang khusus seperti efek-efek khusus. Kemajuan teknologi sudah tidak bisa dibendung lagi, berbagai inovasi pun diciptakan dibidang hiburan. *Game* adalah salah satu inovasi yang tercipta dari perkembangan teknologi.

Saat ini selain *game offline*, sudah muncul berbagai *game online*, yaitu game yang dimainkan secara daring (*online*). Game ini muncul dengan berbagai aliran (*genre*). Banyak orang yang memandang kemunculan *game* sebagai efek negatif dari perkembangan teknologi. Salah satu contohnya adalah posisi tubuh yang kerap kali tidak benar dalam bermain game membuat seseorang mengalami gangguan pertumbuhan (anonim, 2013). Namun juga tidak sedikit yang menganggap kehadiran *game* membawa efek positif dalam dunia teknologi sekarang ini. Jika video *game* digunakan secara baik maka akan menghasilkan efek yang baik pula. Contohnya *game* dapat menurunkan tingkat stress pada diri seseorang (anonim, 2013). Bahkan saat ini *game* sudah bukan lagi hiburan untuk membuang waktu, melainkan sudah berubah menjadi industri baru yang potensial untuk menghasilkan pendapatan yang menggiurkan (Jannah, 2018).

E-sport merupakan permainan video *game* yang kompetitif. *E-sport* memiliki kepanjangan *electronic sport*, yang berarti adalah olahraga elektronik dalam hal ini adalah video *game*. Jika dilihat dua sampai tiga tahun kebelakang e-sport tidak begitu terkenal, namun saat ini *e-sport* begitu banyak dilirik oleh berbagai pihak, dari televisi sampai perusahaan-perusahaan yang bahkan tidak dalam bidang *e-sport*. Posisi *e-sport* sama halnya seperti catur yang sudah

termasuk dalam kategori olahraga dan sudah ada perlombaan di berbagai olimpiade (Restika, 2018). Saat ini ada beberapa aliran (*genre*) *e-sport* yang terkenal, diantaranya adalah MOBA (*Multiplayer Online Battle Arena*), FPS (*First Person Shooter*), Fighting (*Street Fighter*), *Real Time Strategy Games* (Budi, 2018). Sudah banyak kompetisi yang diadakan untuk pertandingan *e-sport* dengan hadiah yang tidak sedikit.

Salah satu kompetisi yang terbaru adalah turnamen *e-sport* Asia Tenggara yang diselenggarakan oleh pemerintah Indonesia dengan total hadiah mencapai Rp1,4 M dan dilaksanakan di Jakarta pada tanggal 17 Oktober sampai 21 Oktober 2018. Game yang dipertandingkan antara lain adalah Mobile Legend, Arena Of Valor (AOV), Playerunknown's Battleground, Point Blank dan DOTA 2. Jika dilihat dari penyelenggara dan dari nominal hadiah yang diperebutkan, maka *e-sport* tidak bisa dipandang rendah. Potensi industri dalam bidang tersebut memang ada dan sudah terlihat.

Salah satu game yang termasuk dalam golongan *e-sport* yang sedang naik daun yaitu *Player Unknown Battleground* atau yang sering disebut dengan PUBG. Game ini termasuk dalam genre permainan FPS (*First Person Shooter*) namun juga dilengkapi dengan mode *Third Person*. Permainan ini memiliki beberapa mode bermain berdasar kategori keanggotaan yaitu solo (sendirian/tanpa tim), *duo* (tim beranggotakan dua orang) dan *squad* (tim beranggotakan empat orang).

Konsep permainan *Player Unknown Battleground* menggunakan konsep *survival*, dimana pemain yang bertahan hidup terakhir maka yang akan menang. Selama pertandingan pemain diminta untuk mencari segala senjata dan perlengkapan yang dibutuhkan untuk bertahan hidup pada suatu pulau. Dalam permainan ini ada istilah *win ratio* yaitu nilai persentase yang menunjukkan rasio seberapa besar ia menang selama ia bermain dari awal ia mendaftarkan akun sampai terakhir ia bermain.

Tidak mudah untuk bertahan hidup dan menjadi yang terakhir dalam pertandingan game tersebut. Terkadang butuh strategi yang matang untuk

memenangkan pertandingan. Selama ini belum ada penelitian terkait pencarian strategi terbaik dalam bermain game ini. Dalam penelitian ini penulis akan menganalisis rekaman data statistik dari berbagai pemain dengan tujuan untuk mencari strategi terbaik dalam bermain game PUBG.

Ada beberapa penelitian yang sudah dilakukan dengan menggunakan metode *Random Forest*. Salah satunya adalah yang dilakukan oleh Nidhomiddin dan Otok (2015). Dalam penelitiannya mereka melakukan klasifikasi penderita HIV/AIDS menggunakan metode *Random Forest* dan *Multivariate Adaptive Regression Spline (MARS) binary response*. Hasil yang didapat adalah mengetahui variabel yang paling berpengaruh untuk menentukan status HIV/AIDS yaitu usia, jenis pekerjaan, pernah ditahan karena kasus NAPZA, status nikah dan selalu pakai jarum steril. Akurasi yang didapatkan ketika menggunakan metode MARS sebesar 80,28%. Sedangkan ketika menggunakan metode *Random Forest* diperoleh akurasi terbaik yaitu 97,80%. Penulis tersebut juga mencoba gabungan antara metode MARS dan *Random Forest*, dan memperoleh hasil akurasi sebesar 91,00%.

Penelitian lain adalah yang dilakukan oleh Nugroho dan Emiliyawati (2017). Dalam penelitiannya mereka melakukan klasifikasi tingkat penerimaan konsumen terhadap mobil menggunakan metode *Random Forest*. Dari hasil penelitian, mereka mendapatkan bahwa variabel yang mempengaruhi tingkat penerimaan konsumen terdiri dari harga pembelian, biaya perawatan, jumlah pintu, kapasitas penumpang, ukuran bagasi dan taksiran keselamatan penumpang.

Berdasar latar belakang tersebut, dalam tugas akhir ini penulis akan menggunakan metode *Random Forest* untuk memprediksi *win ratio* terhadap data statistik pemain game *Player Unknown Battleground* dengan menggunakan metode *Random Forest*. Dengan demikian diharapkan dapat dihasilkan model untuk memprediksi *win ratio*. Model tersebut memiliki beberapa *tree* yang setiap *node* merupakan atribut yang dipilih karena

kemampuannya yang baik untuk mengklasifikasi. Atribut yang dijadikan *Node* pada *tree* dapat dipertimbangkan sebagai strategi dalam bermain *game* PUBG.

1.2. Rumusan Masalah

- a) Bagaimana menerapkan metode *Random Forest* untuk memprediksi nilai *win ratio* dengan baik?
- b) Atribut (feature) apa yang paling berpengaruh dalam memprediksi nilai *win ratio*?
- c) Berapa akurasi prediksi terbaik yang dihasilkan oleh metode *Random Forest* terhadap data statistik pemain game *Player Unknown Battleground*?

1.3. Tujuan

- a) Menerapkan metode *Random Forest* untuk memprediksi nilai *win ratio*.
- b) Mengetahui atribut (feature) yang paling berpengaruh dalam memprediksi nilai *win ratio* yang tinggi.
- c) Menguji akurasi prediksi yang dihasilkan oleh metode *Random Forest* terhadap data statistik pemain game *Player Unknown Battleground*

1.4. Batasan Masalah

Dalam penelitian ini, masalah yang dibatasi adalah:

- a) Merancang model dengan menggunakan metode *Random Forest* untuk prediksi nilai *win ratio* setiap pemain *Player Unknown Battleground*.
- b) Data statistik pemain yang digunakan hanya data dengan mode solo (sendirian/tanpa tim) yang terdiri dari 87.898 baris data dan 52 kolom (atribut). Data tersebut diperoleh dari website Kaggle, yang merupakan salah satu website yang menyediakan data public.
- c) Metode yang digunakan adalah *Random Forest*.

1.5. Sistematika Penulisan

Berikut adalah metodologi penelitian dan langkah-langkah yang digunakan dalam pelaksanaan penelitian ini:

- a) BAB I Pendahuluan
Dalam bab ini diuraikan tentang latar belakang masalah, rumusan masalah, tujuan dan sistematika penulisan.
- b) BAB II Tinjauan Pustaka
Pada bab ini diuraikan landasan teori dan masalah yang berhubungan dengan metode data mining khususnya metode *Random Forest* dengan penelitian terkait.
- c) BAB III Metodologi penelitian
Dalam bab ini akan dijelaskan data dan rencana langkah-langkah yang akan dilakukan dalam melakukan penelitian ini.
- d) BAB IV Analisis dan Desain
Pada bab ini akan dijelaskan bagaimana proses penerapan/implementasi dan analisis dari penerapan metode *Random Forest* untuk mencari strategi yang baik dalam bermain PUBG. Juga memaparkan desain model yang akan digunakan dalam penelitian ini.
- e) BAB V Implementasi dan Analisis Hasil
Pada bab ini akan diuraikan hasil implementasi dari desain yang sudah dirancang dan analisis dari hasil yang didapatkan.
- f) BAB VI Penutup
Pada bab ini akan diuraikan kesimpulan dari hasil percobaan-percobaan yang sudah dilakukan dan juga terdapat saran dari penulis untuk pengembangan dari penelitian ini.
- g) Daftar Pustaka
Berisi tentang daftar referensi dan buku-buku yang digunakan dalam melakukan penelitian ini.

BAB II

TINJAUAN PUSTAKA

2.1. Penambangan Data (*Data Mining*)

Definisi dari penambangan data menurut Han dan Kamber (2012) yaitu “Sebuah penggalian (*mining*) suatu pengetahuan (*knowledge*) dari sekumpulan data yang besar/banyak”. Beberapa orang mengistilahkan *data mining* dengan sebutan lain yaitu *Knowledge Discovery from Data* (KDD). Menurut Han dan Kamber (2012) dalam proses pencarian suatu *knowledge* ada beberapa langkah yang diperlukan, yaitu:

- 1) *Data cleaning*

Pada langkah ini yang dilakukan adalah menghilangkan noise dari data dan data-data yang tidak konsisten.

- 2) *Data integration*

Melakukan kombinasi atau penyatuan data jika memiliki lebih dari satu sumber data.

- 3) *Data selection*

Pemilihan data yang sesuai dan relevan dengan tujuan analisis dari database.

- 4) *Data transformation*

Perubahan/transformasi data ke dalam bentuk yang sesuai untuk dilakukan mining dengan cara melakukan operasi penjumlahan atau agregasi.

- 5) *Data mining*

Proses penting dimana dilakukan penerapan sebuah metode intelijen untuk mencari pola data.

- 6) *Pattern evaluation*

Mengidentifikasi pola yang merepresentasikan sebuah *knowledge* berdasarkan beberapa ukuran menarik (*interestingness measure*).

7) *Knowledge presentation*

Langkah penggunaan teknik visualisasi dan representasi sebuah knowledge yang didapat untuk menyajikan knowledge tersebut ke user.

Secara umum *data mining* dapat diklasifikasi menjadi dua kategori yaitu *descriptive* dan *predictive* (Han, 2012):

- 1) *Descriptive* berperan dalam melakukan perincian yang bersifat umum dari data yang terdapat dari database, biasa digunakan dalam merepresentasikan data.
- 2) *Predictive* berperan dalam memprediksi data yang terdapat dalam database.

Dalam data mining ada beberapa pendekatan yang digunakan untuk mengambil informasi dari sekumpulan data (Han, 2012), yaitu:

- 1) *Supervised learning* (pembelajaran dengan diawasi)
Semua data diberi label dan algoritma belajar untuk memprediksi keluaran dari data masukan.
- 2) *Unsupervised learning* (pembelajaran tanpa diawasi)
Semua data tidak diberi label dan algoritma belajar ke struktur yang melekat dari data input.
- 3) *Semi-supervised learning* (pembelajaran semi diawasi)
Beberapa data diberi label tetapi sebagian besar tidak diberi label dan campuran teknik yang *Supervised learning* dan *Unsupervised learning* dapat digunakan.

2.2. Metode Random Forest

Random Forest merupakan sebuah metode *ensemble*. Metode *ensemble* merupakan cara untuk meningkatkan akurasi metode klasifikasi dengan cara mengkombinasikan metode klasifikasi (Han, 2012). *Random Forest* diawali dengan teknik dasar *data mining* yaitu *decision tree*. Pada *decision tree* input dimasukan pada bagian atas (*root*) kemudian turun kebagian bawah (*leaf*) untuk menentukan data tersebut termasuk kelas apa. *Random forest* adalah pengklasifikasi yang terdiri dari kumpulan pengklasifikasi pohon terstruktur dimana masing-masing pohon melemparkan unit suara untuk kelas paling populer di input x (Breiman, 2001). Dengan kata lain *Random Forest* terdiri dari sekumpulan *decision tree* (pohon keputusan), dimana kumpulan *decision tree* tersebut digunakan untuk mengklasifikasi data ke suatu kelas.

Random forest merupakan metode klasifikasi yang *supervised*. Sesuai dengan namanya, metode ini menciptakan sebuah hutan (*forest*) dengan sejumlah pohon (*tree*). Secara umum, semakin banyak pohon (*tree*) pada sebuah hutan (*forest*) maka semakin kuat juga hutan tersebut terlihat. Pada kasus yang sama, semakin banyak *tree*, maka semakin besar pula akurasi yang didapatkan (Polamuri, 2017).

Decision Tree akan menggunakan *information gain* dan *gini index* untuk perhitungan dalam menentukan *root node* dan *rule*. Sama halnya dengan *Random Forest* yang akan menggunakan *information gain* dan *gini index* untuk perhitungan dalam membangun *tree* (Han, 2012), hanya saja *Random Forest* akan membangun lebih dari satu *tree*. Masing-masing *tree* dibangun menggunakan set data dengan atribut yang diambil secara acak dari *data training*. Dengan kata lain setiap *tree* akan bergantung pada nilai dari sampel vektor yang independen dengan distribusi yang sama pada setiap *tree* (Han, 2012). Selama proses klasifikasi setiap *tree* akan memberikan *voting* kelas yang paling populer (Han, 2012).

Cara kerja dari *Random Forest* dapat digambarkan dengan kasus nyata dalam kehidupan nyata yang dianalogikan seperti kasus berikut ini (Polamuri, 2017). Seorang mahasiswa ingin berlibur ke suatu tempat tapi ia tidak tahu ingin berlibur kemana. Mahasiswa tersebut memutuskan untuk bertanya pada satu orang sahabatnya. Kemudian sahabatnya tersebut memberikan beberapa pertanyaan untuk memutuskan rekomendasi tempat yang cocok untuk mahasiswa tersebut. Sejauh ini kasus tersebut menggambarkan metode *Decision Tree*, dimana seorang sahabat dari mahasiswa tersebut menggambarkan *tree* yang dibangun untuk memutuskan rekomendasi tempat. Kemudian mahasiswa tersebut memutuskan lagi untuk bertanya ke teman yang lain dengan jumlah yang banyak dan setiap orang mengajukan pertanyaan-pertanyaan yang acak, setiap teman tentu akan memberikan rekomendasi yang berbeda dan juga ada yang sama. Lalu mahasiswa tersebut akan memutuskan untuk pergi ke tempat yang paling banyak direkomendasikan dari teman-temannya. Inilah yang menggambarkan proses dan cara kerja metode *Random Forest*, dimana akan membentuk *tree* yang banyak untuk memutuskan suatu keputusan, lalu keputusan akhir akan ditentukan oleh hasil keputusan terbanyak dari *tree* yang telah di bangun. Konsep *voting* seperti ini disebut dengan *majority voting*.

Setelah menggunakan analogi kasus kehidupan nyata, berikut ini adalah contoh kasus lain. Gambar 2.1 menggambarkan sebuah *dataset* yang akan dilakukan klasifikasi menggunakan metode *Random Forest*. Kemudian dari *dataset* tersebut dibagi menjadi tiga bagian secara acak (*random*) yang digambarkan pada gambar 2.2, gambar 2.3 dan gambar 2.4. Dari masing-masing data yang telah dibagi secara acak tersebut akan dibangun *tree*, maka jumlah *tree* yang terbangun adalah tiga. Dari ketiga *tree* tersebutlah yang akan digunakan untuk memutuskan suatu data akan diklasifikasi ke suatu kelas tertentu menggunakan konsep *majority voting*.

f11	f12	f13	f14	f15	t1
f21	f22	f23	f24	f25	t2
f31	f32	f33	f34	f35	t3
:	:	:	:	:	:
:	:	:	:	:	:
fm1	fm2	fm3	fm4	fm5	tm

Gambar 2. 1 *Dataset*

sumber: dataaspirant.com

f11	f12	f13	f14	f15	t1
f81	f82	f83	f84	f85	t8
f71	f72	f73	f74	f75	t7
:	:	:	:	:	:
:	:	:	:	:	:
fm1	fm2	fm3	fm4	fm5	tm

Gambar 2. 2 *Random Dataset* untuk *tree 1*

sumber: dataaspirant.com

f11	f12	f13	f14	f15	t1
f31	f32	f33	f34	f35	t3
f41	f42	f43	f44	f45	t4
:	:	:	:	:	:
:	:	:	:	:	:
fm1	fm2	fm3	fm4	fm5	tm

Gambar 2. 3 *Random Dataset* untuk *tree 2*

sumber: dataaspirant.com

f31	f32	f12	f34	f15	t1
f61	f62	f63	f64	f65	t6
f91	f92	f73	f94	f95	t9
:	:	:	:	:	:
:	:	:	:	:	:
fm1	fm2	fm3	fm4	fm5	tm

Gambar 2. 4 *Random Dataset* untuk *tree 3*

sumber: dataaspirant.com

Berikut ini adalah algoritma dari metode *random forest*. Algoritma dibagi menjadi dua bagian, bagian pertama adalah pembuatan “n” pohon (*tree*) untuk membentuk hutan (*forest*) yang acak (*random*). Bagian kedua adalah algoritma untuk melakukan prediksi dari *Random Forest* yang sudah dibuat (Han, 2012).

Input:

- D , *dataset* yang terdiri dari d baris
- k , angka dari jumlah *tree*

Langkah-langkah metode *Random Forest* (Bagian 1):

1. Buat data sampel data D_i dengan mengambil acak dari *dataset* D dengan pengembalian (*replacement*)
2. Gunakan sampel data D_i untuk membangun *tree* ke i ($i = 1, 2, \dots k$)
3. Ulangi langkah satu dan dua sebanyak k

Metode *Random Forest* diawali dengan pemilihan “ k ” sampel *dataset* D_i yang diambil secara acak dengan pengembalian (*replacement*). Langkah selanjutnya adalah menggunakan *dataset* D_i untuk membangun *decision tree* ke- i . Dalam membangun *tree* ke- i , metodologi CART dapat digunakan. Metodologi CART menggunakan *information gain* dalam menentukan setiap *node* pada *tree*. Perhitungan *information gain* dapat dihitung menggunakan rumus 2.1.

Pseudocode untuk proses prediksi *data test* (Bagian 2):

- 1) Ambil *data test* dan gunakan rule dari setiap *tree* untuk memprediksi keluaran klasifikasi dari data tersebut, simpan hasil yang didapat
- 2) Hitung suara (*vote*) untuk setiap target yang diprediksi dari setiap *tree*

- 3) Pertimbangkan target prediksi yang terpilih dengan memilih target kelas yang paling banyak diprediksi sebagai hasil prediksi akhir dari metode *random forest*

Untuk memprediksi kelas target dari *data test* menggunakan *random forest*, masukan data test melalui aturan-aturan (*rule*) yang sudah dibuat menggunakan *tree*. Hasil prediksi setiap *tree* bisa saja ada yang berbeda dan ada yang sama, maka prediksi akhir akan dipilih berdasar prediksi kelas yang terbanyak diprediksi. Misalkan dari 100 *tree*, 80 *tree* memprediksi target adalah kelas A dan sisanya adalah kelas B, maka prediksi akhir yang dipilih adalah kelas A konsep pemilihan dengan suara terbanyak yaitu hasil prediksi dari semua *tree* dinamakan *majority voting*.

2.3. Classification And Regression Tree (CART)

Pembangunan *tree* dalam metode *Random Forest* menggunakan metodologi yang sama yang digunakan oleh *Classification And Regression Tree* (CART) (Han, 2012). CART menggunakan *information gain* untuk mengukur pemilihan atribut yang akan digunakan pada setiap *node* sebuah *tree*. Katakanlah N adalah node yang akan digunakan untuk memisahkan setiap kelas dengan menggunakan atribut dari *dataset D*. Atribut dengan *information gain* yang tertinggi akan digunakan untuk memisahkan (*split*) node N . Penghitungan nilai *information gain* dapat dicari menggunakan rumus sebagai berikut:

$$Gain(A) = Info(D) - Info_A(D) \dots \dots \dots (2.1)$$

Dimana nilai $Info(D)$ dapat dicari menggunakan rumus 2.2 dan $Info_A(D)$ dicari menggunakan rumus 2.3 dibawah ini:

$$Info(D) = -\sum_{i=1}^m p_i \log_2(p_i) \dots \dots \dots (2.2)$$

Keterangan:

m : jumlah kelas target

p_i : probabilitas munculnya kelas ke i pada partisi D

$$Info_A(D) = \sum_{j=1}^v \frac{|D_j|}{|D|} \times Info(D_j) \dots \dots \dots (2.3)$$

Keterangan:

v : jumlah partisi

D_j : total partisi ke j

D : jumlah *tuple*/baris pada semua partisi

Untuk menghitung *information gain* dari atribut yang memiliki nilai kontinu (numerik), maka harus ditentukan nilai pembelah (*split-point*) terbaik untuk mengelompokan nilai dari atribut tersebut. Untuk mencari *split-point* terbaik maka data dari atribut tersebut harus diurutkan terlebih dahulu. Nilai tengah antara setiap pasangan nilai yang berdekatan dianggap sebagai kemungkinan yang bisa dijadikan *split-point* (Han, 2012). Jika sebuah atribut A adalah atribut yang memiliki nilai kontinu, maka semua nilai A diurutkan, lalu langkah selanjutnya adalah mencari titik tengahnya, maka kemungkinan jumlah partisi adalah dua dengan $v = 2$ ($j = 1$ dan 2) pada persamaan 2.3.

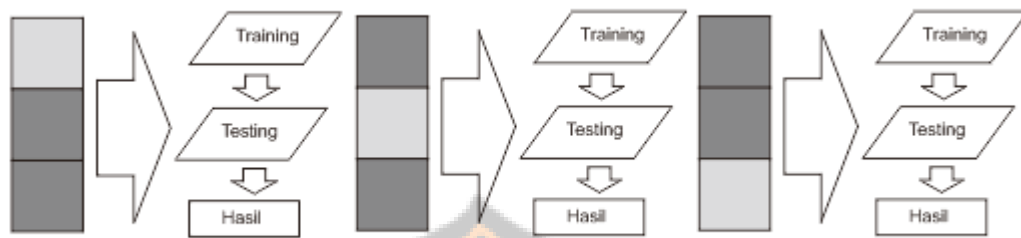
2.4. Evaluasi Kerja

Sesudah hasil prediksi didapatkan dengan menggunakan pemodelan yang sudah dibangun. Maka proses selanjutnya adalah melakukan evaluasi kinerja pemodelan tersebut untuk mengetahui seberapa akurat prediksi yang didapatkan. Berikut adalah teknik untuk melakukan evaluasi kinerja (Han 2006):

2.4.1. *K-Fold Cross Validation*

Metode *Cross Validation* merupakan salah satu metode untuk mengevaluasi algoritma *learning* dengan membagi data menjadi dua segmen yaitu. Segmen pertama digunakan untuk pembelajaran (*learning*) pelatihan (*training*) model, segmen kedua digunakan untuk validasi model. Ciri khas dari *Cross Validation* adalah set *training* dan validasi harus disilangkan (cross-over) dalam putaran berturut-turut sehingga setiap titik data memiliki peluang untuk divalidasi. Dasar dari *Cross Validation* adalah *K-Fold Cross Validation* (Refaeilzadeh, 2008).

Dalam *K-Fold Cross Validation*, langkah pertama kali adalah data akan dipartisi ke dalam segmen atau *fold* yang sama atau identik (nyaris sama). Berikutnya adalah iterasi ke k dari *training* dan validasi dilakukan sedemikian rupa sehingga dalam setiap iterasi *fold* data yang berbeda dimunculkan (*held-out*) untuk validasi, sementara sisa *fold* $k-1$ digunakan untuk *training*. Pada gambar 2.1 menggambarkan contoh dengan $k = 3$. Bagian yang gelap menggambarkan data untuk *training* sedangkan bagian terang menggambarkan data untuk validasi (*testing*). Dalam *data mining* maupun *machine learning* 10-fold cross-validation ($k=10$) merupakan yang paling umum atau sering digunakan (Refaeilzadeh, 2008).



Gambar 2.5 Prosedur 3-Fold Validation

Terdapat dua kemungkinan tujuan yang bisa didapatkan dari *Cross Validation*, yaitu:

- Untuk mengestimasi performa suatu pemodelan yang menggunakan metode untuk pembelajaran (*learned*) suatu data. Dengan kata lain untuk mengukur generalisasi metode.
- Untuk membandingkan performa dari dua atau lebih metode yang berbeda dan menemukan metode yang terbaik untuk suatu data, atau untuk membandingkan performa dari dua atau lebih variasi parameter yang digunakan dalam pemodelan.

2.4.2. *Confusion Matrix*

Confusion matrix digunakan untuk melihat seberapa baik atau seberapa besar akurasi yang dihasilkan dari model klasifikasi yang sudah dibuat untuk memprediksi atau mengklasifikasi kelas dari data *testing*. Rincian hasil klasifikasi berupa prediksi kelas ditampilkan di atas dan kelas yang aktual di bawah kiri, lihat Tabel 2.1.

Tabel 2. 1 Matrik Klasifikasi untuk Model dua kelas

	Kelas Prediksi		
		Kelas=Positif	Kelas=Negatif
	Kelas=Positif	TP (True Positive)	FN (False Negative)
Kelas aktual	Kelas=Negatif	FP (False Positive)	TN (True Negative)

Nilai akurasi dapat dihitung dengan menggunakan rumus (2.5) yang didefinisikan seperti berikut:

$$Akurasi = \frac{TP+TN}{TP+TN+FP+FN} \dots\dots\dots (2.5)$$

dimana:

TP : Jumlah kelas positif yang diklasifikasi sebagai positif

FP : Jumlah kelas negatif yang diklasifikasi sebagai positif

TN : Jumlah kelas negatif yang diklasifikasi sebagai negatif

FN : Jumlah kelas negatif yang diklasifikasi sebagai negatif

BAB III

METODOLOGI PENELITIAN

3.1. Bahan Penelitian/Data

Data yang digunakan dalam penelitian ini adalah data statistik pemain *Player Unknown Battleground* yang didapatkan dari *website* Kaggle. Data tersebut memiliki 87.898 *record* (baris) dan 152 atribut (kolom) dengan gabungan mode permainan *solo*, *duo* dan *squad*. Data ini diunggah oleh Justin Moore pada tahun 2017 dengan ekstensi *.csv*. Pada tabel 3.1 berikut adalah penjelasan dari masing-masing atribut beserta contoh data dengan mode solo sejumlah 52 atribut. Sebagai catatan untuk mempermudah penulisan tabel maka nama atribut diletakan di sebelah kiri tabel.

Tabel 3. 1 Penjelasan Atribut dan Contoh Data Statistik Pemain PUBG

No	Nama Atribut	Keterangan	Contoh Data	Record/ Row (Baris data ke)
1	player_name	Nama pemilik akun	Blackwalk	2
2	tracker_id	<i>Id tracker</i> pemilik akun	8199	2
3	solo_WinRatio	Rasio kemenangan	18,18	2
4	solo_killDeathRatio	Ratio jumlah korban	4,41	2
5	solo_TimeSurvived	total waktu bertahan hidup dalam satuan detik	33014,86	2
6	solo_RoundsPlayed	Total ronde bermain	33	2
7	solo_Wins	Total kemenangan	6	2

8	solo_WinTop10Ratio	Rasio kemenangan posisi pertama dan kemenangan 10 besar	0,36	2
9	solo_Top10s	Total kemenangan 10 besar	11	2
10	solo_Top10Ratio	Rasio kemenangan dalam 10 besar	33,3	2
11	solo_Losses	Jumlah kekalahan	27	2
12	solo_Rating	Rating yang dimiliki terakhir kali	1884,53	2
13	solo_BestRating	Rating terbaik selama bermain	1860,74	2
14	solo_DamagePg	Jumlah <i>damage</i> yang diciptakan per ronde	393,04	2
15	solo_HeadshotKillsPg	Jumlah <i>head shot</i> yang diciptakan per ronde	1,27	2
16	solo_HealsPg	Jumlah poin <i>heal</i> (penyembuhan) per ronde	1,82	2
17	solo_KillsPg	Jumlah korban per ronde	3,61	2
18	solo_MoveDistancePg	Jarak yang ditempuh per ronde	5021,41	2
19	solo_RoadKillsPg	Rata-rata korban yang dibunuh menggunakan kendaraan	0,06	2

20	solo_TeamKillsPg	Jumlah anggota tim yang dibunuh per ronde	0	2
21	solo_TimeSurvivedPg	Waktu bertahan hidup per ronde dalam satuan detik	1000,45	2
22	solo_Top10sPg	Rata-rata kemenangan di sepuluh besar	0,33	2
23	solo_Kills	Jumlah korban	119	2
24	solo_Assists	Jumlah bantuan	2	2
25	solo_Suicides	Jumlah kejadian bunuh diri (mati tanpa diserang musuh)	0	2
26	solo_TeamKills	Jumlah anggota tim yang dibunuh	0	2
27	solo_HeadshotKills	Jumlah <i>head shot</i> (tembakan tepat kepala)	42	2
29	solo_HeadshotKillRatio	Rasio <i>head shot</i> (tembakan tepat kepala)	0,35	2
30	solo_VehicleDestroys	Jumlah penghancuran kendaraan	3	2
31	solo_RoadKills	Jumlah korban yang dibunuh menggunakan kendaraan	2	2
32	solo_DailyKills	Jumlah korban harian (rata-rata korban dalam satu hari)	18	2

33	solo_WeeklyKills	Jumlah korban minggu (rata-rata korban dalam satu minggu)	18	2
34	solo_RoundMostKills	Jumlah korban terbanyak dalam satu rounde	13	2
35	solo_MaxKillStreaks	Jumlah korban terbanyak yang diciptakan berturut-turut	3	2
36	solo_Days	Jumlah hari bermain	10	2
37	solo_LongestTimeSurvived	Waktu terlama bertahan hidup	1987,94	2
38	solo_MostSurvivalTime	Waktu terlama bertahan hidup	1987,94	2
39	solo_AvgSurvivalTime	Rata-rata waktu bertahan hidup	1221,32	2
40	solo_WinPoints	Jumlah poin kemenangan	3812	2
41	solo_WalkDistance	Jumlah jarak yang sudah ditempuh dengan jalan kaki	47868,77	2
42	solo_RideDistance+AP1	Jumlah jarak yang sudah ditempuh dengan kendaraan	117837,7	2
43	solo_MoveDistance	Jumlah jarak pergerakan yang sudah ditempuh	165706,5	2
44	solo_AvgWalkDistance	Rata-rata jarak yang sudah ditempuh dengan jalan kaki	2017,38	2
45	solo_AvgRideDistance	Rata-rata jarak yang sudah ditempuh dengan kendaraan	5188,69	2

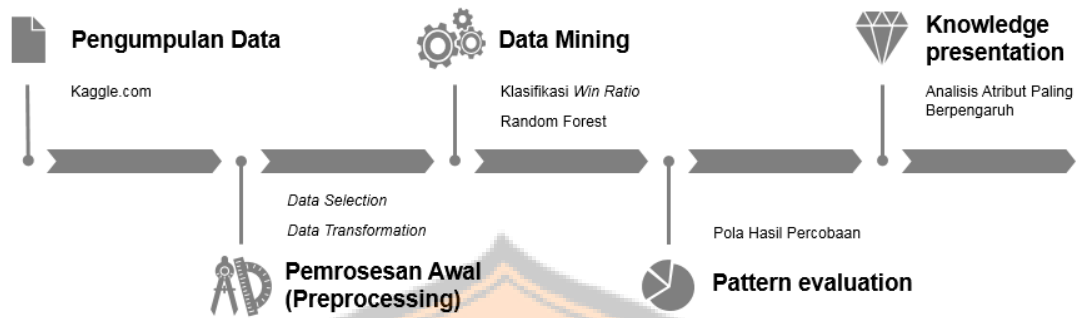
46	solo_LongestKill	Jarak korban terjauh	351,95	2
47	solo_Heals	Jumlah poin <i>heal</i> (penyembuhan)	60	2
48	solo_Boosts	Total energi yang digunakan	88	2
49	solo_DamageDealt	Jumlah poin <i>damage</i> (kerusakan) yang diterima	12970,39	2
50	solo_Revives	Jumlah kejadian penyelamatan teman satu tim	0	2
51	solo_WeaponAcquired	Senjata yang di dapatkan	0	2
52	solo_DBNOs	Singkatan dari 'Down But No Out' yang merupakan kejadian dimana <i>Health Points</i> (HP) habis namun tetap hidup	0	2

3.2. Peralatan Penelitian

Adapun peralatan yang digunakan pada penelitian ini adalah Laptop Asus dengan spesifikasi processor Intel(R) Pentium(R) CPU B980 2,40GHz, RAM 4GB dan Harddisk 500GB. Aplikasi yang digunakan dalam penelitian ini untuk pembuatan model adalah aplikasi Netbeans yang merupakan *Integrated Development Environment* (IDE) yang berbasis bahasa *java*.

3.3. Tahap-Tahap Penelitian

Secara singkat langkah-langkah dalam penelitian ini digambarkan pada skema penelitian yang digambarkan pada Gambar 3.1. Penjelasan lebih detail akan dijelaskan pada 3.1.1 sampai 3.1.5.



Gambar 3. 1 Skema penelitian

3.1.1. Pengumpulan Data

Tahap awal dalam penelitian ini adalah pengumpulan data. Pengumpulan data yang dilakukan dalam penelitian ini dengan cara mengunduh data dari *website* Kaggle.com.

3.1.2. Penelitian Pustaka

Studi literatur dilakukan oleh penulis untuk mendapatkan referensi yang berkaitan dengan teori yang dapat mendukung penelitian ini. Literatur yang didapatkan berasal dari buku dan karya ilmiah.

3.1.3. *Knowledge Discovery in Database (KDD)*

Dengan mengacu proses KDD menurut Han (2006), berikut ini adalah proses KDD yang dilakukan pada penelitian ini:

a) *Data selection*

Proses *data selection* merupakan proses pemilihan atau penghapusan data atau atribut yang tidak digunakan pada *dataset*. Pada tahap ini penulis melakukan secara manual menggunakan aplikasi microsoft excel.

b) *Data transformation*

Proses *data transformation* merupakan proses perubahan data kedalam bentuk yang berbeda dari bentuk sebelumnya untuk mendukung proses penggalian informasi data (*data mining*). Pada tahap ini penulis juga melakukan secara manual menggunakan aplikasi microsoft excel.

c) *Data mining*

Proses *data mining* adalah proses penggalian informasi dari data yang ada. Penggalian informasi pada penelitian ini berupa klasifikasi. Dalam penelitian ini proses klasifikasi akan dilakukan menggunakan metode *Random Forest* untuk data statistik pemain PUBG.

d) *Pattern evaluation dan Knowledge presentation*

Untuk proses mengidentifikasi pola yang tepat yang merupakan hasil dari proses *data mining* (*Pattern evaluation*) dan proses penyajian hasil dari *data mining* kepada user (*Knowledge presentation*) dilakukan setelah sistem selesai dibangun dan proses penambahan data selesai dilakukan. Dalam proses ini penulis melakukan evaluasi dari hasil penambahan data yang didapat dari perangkat lunak yang telah dibangun dan menjelaskan hasil evaluasi tersebut agar informasi yang didapat dapat dengan mudah diterima oleh pihak-pihak yang membutuhkan.

3.1.4. Pengembangan Perangkat Lunak

Penulis melakukan pengembangan perangkat lunak sebagai alat untuk mengolah *dataset* yang dimiliki penulis untuk mendapatkan informasi (*knowledge*) yang berguna. Metode yang digunakan oleh penulis adalah metode *waterfall*. Metode tersebut adalah salah satu

metode yang sangat sering digunakan oleh para pengembang perangkat lunak. Pengerjaan sistem secara *linear* diberlakukan dalam metode ini. Dimana jika tahap pertama belum selesai maka tahap kedua belum bisa dilakukan. Secara garis besar metode *waterfall* memiliki langkah-langkah sebagai berikut:

1) Analisis

Pada langkah ini yang dilakukan adalah melakukan analisis kebutuhan-kebutuhan sistem. Pengumpulan data dapat berupa sebuah penelitian, wawancara atau studi literatur.

2) Desain

Pada tahap ini merupakan proses penerjemahan kebutuhan-kebutuhan yang sudah didapatkan pada tahap sebelumnya ke sebuah perancangan perangkat lunak yang dapat diperkirakan sebelum masuk ke tahap implementasi. Pada tahap ini menghasilkan dokumen yang disebut *software requirement*. Dokumen tersebutlah yang akan digunakan oleh programmer untuk membangun atau mengimplementasi sistemnya.

3) Implementasi

Tahap ini merupakan tahap penerapan desain ke dalam bahasa pemrograman. Dalam tahap ini programmer akan mengubah proses transaksi yang diinginkan *user* ke dalam sistem yang akan dibuat

4) Pengujian Perangkat Lunak

Tahap selanjutnya adalah tahap pengujian terhadap perangkat lunak yang sudah di bangun. Model pengujian yang dilakukan oleh penulis adalah pengujian *black box* dan pengujian membandingkan hasil perhitungan manual dengan hasil yang diperoleh oleh perangkat lunak. Tujuannya adalah untuk memeriksa

apakah ada kesalahan-kesalahan yang terdapat pada perangkat lunak tersebut agar dapat diperbaiki.

3.1.5. Analisis dan Pembuatan Laporan

Dalam tahap ini dilakukan analisa kinerja terhadap model atau metode *Random Forest* yang sudah diimplementasikan ke dalam sebuah perangkat lunak. Hasil dari analisa tersebut akan dicatat dalam laporan tugas akhir.



BAB IV

PEMROSESAN AWAL DAN PERANCANGAN SISTEM

4.1. Pemrosesan Awal

4.1.1. Seleksi data (*Data Selection*)

Dalam tahap seleksi data akan dilakukan pemilihan atribut yang relevan untuk digunakan dalam penelitian. Atribut yang tidak digunakan atau tidak relevan akan dihapus. Atribut yang akan dihapus dari data statistik Pemain *Player Unknown Battleground* adalah atribut yang tidak independen. Berikut ini adalah daftar atribut yang dihapus beserta alasannya yang ditunjukkan pada tabel 4.1. Dengan demikian didapatkan atribut yang akan digunakan pada tabel 4.2. Sebagai catatan, atribut `solo_RoundsPlayed` akan disimpan untuk pengujian perangkat lunak menggunakan *dataset*.

Tabel 4. 1 Daftar atribut yang dihapus

No	Nama Atribut	Alasan Penghapusan
1	<code>player_name</code>	Memiliki nilai unik (berbeda pada setiap baris data)
2	<code>tracker_id</code>	Memiliki nilai unik (berbeda pada setiap baris data)
3	<code>solo_RoundsPlayed</code>	Digunakan untuk menghitung atribut <code>solo_WinRatio</code>
4	<code>solo_Wins</code>	Digunakan untuk menghitung atribut <code>solo_WinRatio</code>
5	<code>solo_TimeSurvived</code>	Digunakan untuk menghitung atribut <code>solo_TimeSurvivedPg</code>
6	<code>solo_Top10s</code>	Digunakan untuk menghitung atribut <code>solo_WinTop10Ratio</code>
7	<code>solo_Kills</code>	Digunakan untuk menghitung atribut <code>solo_killDeathRatio</code>

8	solo_Assists	Tidak memiliki nilai untuk permainan mode solo
9	solo_TeamKills	Tidak memiliki arti untuk permainan mode solo
10	solo_TeamKillsPg	Tidak memiliki arti untuk permainan mode solo
11	solo_HeadshotKills	Digunakan untuk menghitung atribut solo_HeadshotKillsPg
12	solo_RoadKills	Digunakan untuk menghitung atribut solo_RoadKillsPg
13	solo_Days	Digunakan untuk menghitung atribut solo_DailyKills
14	solo_MostSurvivalTime	Memiliki nilai yang sama dengan atribut solo_LongestTimeSurvived
15	solo_WalkDistance	Digunakan untuk menghitung atribut solo_AvgWalkDistance
16	solo_MoveDistance	Digunakan untuk menghitung atribut
17	solo_RideDistance+AP1	Digunakan untuk menghitung atribut solo_AvgRideDistance
18	solo_Revives	Tidak memiliki arti untuk permainan mode solo (menyelamatkan teman satu tim)
19	solo_WeaponAcquired	Memiliki nilai nol untuk semua baris data
20	solo_Assists	Tidak memiliki arti untuk permainan mode solo (membantu teman satu tim)
21	solo_DBNOs	Memiliki nilai nol untuk semua baris data
22	Solo_WinTop10Ratio	Menggunakan atribut solo_WinRatio (atribut label/kelas) untuk menghitung nilai

Tabel 4. 2 Tabel Atribut Data Statistik Pemain *Player Unknown Battleground* Yang Akan Digunakan

No	Nama Atribut	Keterangan
1	solo_WinRatio	Rasio kemenangan
2	solo_killDeathRatio	Ratio jumlah korban
3	solo_Top10Ratio	Rasio kemenangan dalam 10 besar
4	solo_Losses	Jumlah kekalahan
5	solo_Rating	Rating yang dimiliki terakhir kali
6	solo_BestRating	Rating terbaik selama bermain
7	solo_DamagePg	Jumlah <i>damage</i> yang diciptakan per ronde
8	solo_HeadshotKillsPg	Jumlah <i>head shot</i> yang diciptakan per ronde
9	solo_HealsPg	Jumlah poin <i>heal</i> (penyembuhan) per ronde
10	solo_KillsPg	Jumlah korban per ronde
11	solo_MoveDistancePg	Jarak yang ditempuh per ronde
12	solo_RoadKillsPg	Rata-rata korban yang dibunuh menggunakan kendaraan
13	solo_TimeSurvivedPg	Waktu bertahan hidup per ronde
14	solo_Top10sPg	Rata-rata kemenangan di sepuluh besar
15	solo_Suicides	Jumlah kejadian bunuh diri (mati tanpa diserang musuh)
16	solo_HeadshotKillRatio	Rasio <i>head shot</i> (korban tepat kepala)
17	solo_VehicleDestroys	Jumlah penghancuran kendaraan
18	solo_DailyKills	Jumlah korban harian (rata-rata korban dalam satu hari)
19	solo_WeeklyKills	Jumlah korban minggu (rata-rata korban dalam satu minggu)

20	solo_MaxKillStreaks	Jumlah korban terbanyak yang diciptakan berturut-turut
21	solo_LongestTimeSurvived	Waktu terlama bertahan hidup
22	solo_AvgSurvivalTime	Rata-rata waktu bertahan hidup
23	solo_WinPoints	Jumlah poin kemenangan terakhir
24	solo_AvgWalkDistance	Rata-rata jarak yang sudah ditempuh dengan jalan kaki
25	solo_AvgRideDistance	Rata-rata jarak yang sudah ditempuh dengan kendaraan
26	solo_LongestKill	Jarak korban terjauh
27	solo_Heals	Jumlah poin <i>heal</i> (penyembuhan)
28	solo_Boosts	Total energi yang digunakan
29	solo_DamageDealt	Jumlah poin <i>damage</i> (kerusakan) yang diterima
30	solo_RoundMostKills	Jumlah korban terbanyak dalam satu ronde

4.1.2. Transformasi data (*Data Transformation*)

Ada beberapa transformasi data yang dilakukan penulis pada *dataset* statistik Pemain *Player Unknown Battleground*. Transformasi yang pertama adalah merubah nilai atribut *solo_WinRatio* dari nilai yang kontinu menjadi kategorial dengan cara mengelompokkan nilai tersebut menjadi tiga kelompok. Dalam mengelompokkan nilai atribut *solo_WinRatio*, penulis menggunakan *tools* Weka versi 3.8. Metode yang penulis pilih adalah metode simple k means dengan hasil nilai centroid akhir atribut *solo_WinRatio* yang ditunjukkan pada gambar 4.1.

Selain atribut *solo_WinRatio*, atribut *solo_RoundsPlayed* juga akan dikelompokkan (*clustering*) menjadi 3 kelompok yang menggambarkan tipe pemain yaitu pemula, normal, profesional. Cara pengelompokannya pun sama

seperti mengelompokan atribut `solo_WinRatio`, yaitu menggunakan *tools* Weka versi 3.8 dengan metode simple k means. Hasil nilai akhir centroid atribut `solo_RoundsPlayed` ditunjukkan pada gambar 4.2. Pada gambar 4.3 merupakan hasil klustering untuk atribut `solo_WinRatio` setelah menghilangkan kelompok pemain dengan kluster 0 (pemula) berdasar atribut `solo_RoundsPlayed` (gambar 4.2).

Final cluster centroids:

Attribute	Full Data	Cluster#	0	1	2
	(87898.0)	(38085.0)	(10199.0)	(39614.0)	
<code>solo_WinRatio</code>	5.0175	2.663	19.1609	3.6398	

Gambar 4. 1 Hasil Klustering *Tools* Weka Untuk Atribut `solo_WinRatio`

Final cluster centroids:

Attribute	Full Data	Cluster#	0	1	2
	(87898.0)	(38225.0)	(10254.0)	(39419.0)	
<code>solo_RoundsPlayed</code>	79.2753	38.2316	10.8726	136.8694	

Gambar 4. 2 Hasil Klustering *Tools* Weka Untuk Atribut `solo_RoundsPlayed`

Final cluster centroids:

Attribute	Full Data	Cluster#	0	1	2
	(58366.0)	(13804.0)	(24153.0)	(20409.0)	
<code>solo_WinRatio</code>	3.0715	6.5579	1.8604	2.1468	

Gambar 4. 3 Hasil Klustering *Tools* Weka Untuk Atribut `solo_WinRatio` Setelah Menghilangkan Kluster 0 atribut `solo_RoundsPlayed`

Transformasi yang kedua adalah dengan menormalisasi seluruh nilai atribut kedalam *range*/jarak yang sama yaitu nol sampai satu. Proses normalisasi dilakukan dengan menggunakan metode *min-max normalization*,

adapun rumus yang digunakan dalam metode tersebut dideskripsikan pada rumus 4.1 berikut:

$$v' = \frac{v - \min_A}{\max_A - \min_A} (new_{\max_A} - new_{\min_A}) + new_{\min_A} \dots \dots \dots (4.1)$$

Keterangan:

- v : nilai sebelum ternormalisasi
- v' : nilai setelah ternormalisasi
- \min_A : nilai minimal dari atribut A
- \max_A : nilai maksimal dari atribut A
- new_{\min_A} : nilai minimal terbaru dari atribut A
- new_{\max_A} : nilai maksimal terbaru dari atribut A

4.2. Perancangan Perangkat Lunak

4.2.1. Analisis Kebutuhan Perangkat Lunak

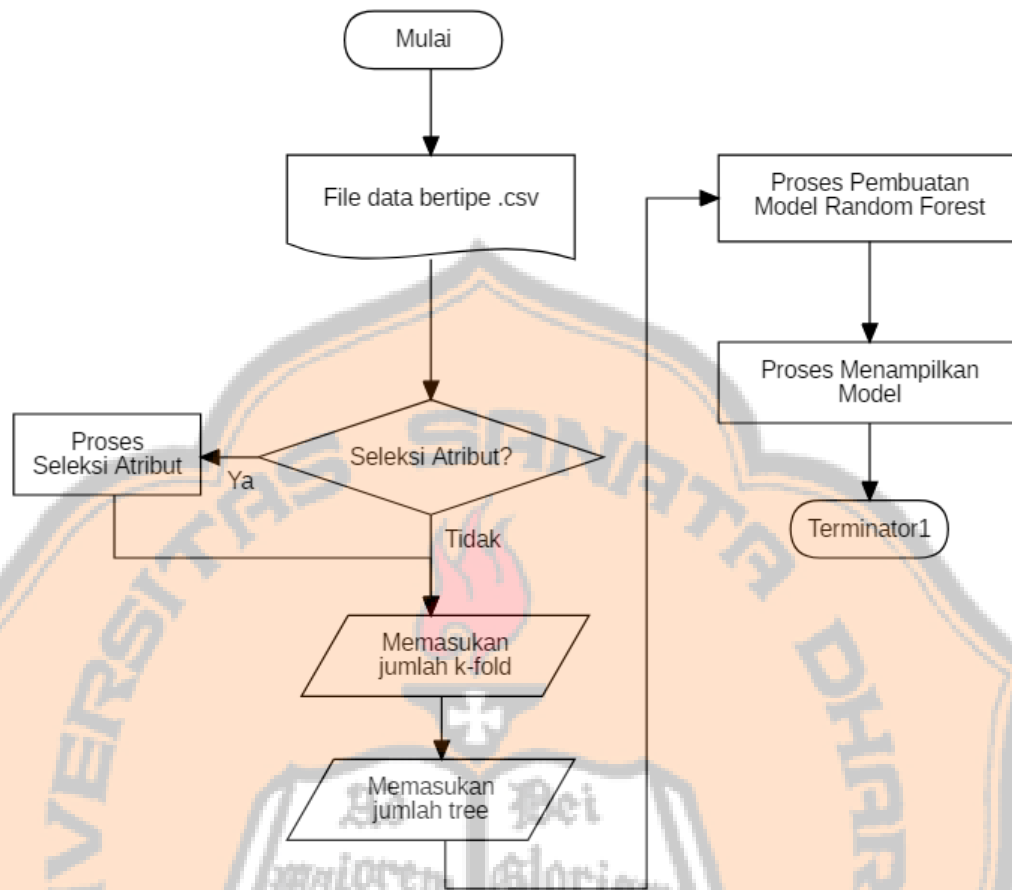
4.2.1.1. Input Sistem

Sistem ini membutuhkan beberapa masukan (*input*) sebelum masuk pada proses klasifikasi. Masukan yang dibutuhkan yaitu:

- a. *File* yang berekstensi .csv
- b. Jumlah *tree*
- c. Jumlah *k-fold*

4.2.1.2. Proses Sistem

Adapun proses pada sistem yang dibangun ini memiliki beberapa tahapan untuk membangun model klasifikasi *random forest*. Tahapan-tahapan tersebut digambarkan pada gambar 4.4.

Gambar 4. 4 Diagram *Flowchart*

4.2.1.3. *Output Sistem*

Keluaran dari sistem ini berupa model klasifikasi *random forest* yang terdiri dari beberapa *tree* yang divisualisasikan sedemikian rupa agar dapat mudah dibaca oleh pengguna. Contoh hasil visualisasi *tree* dapat dilihat pada gambar 4.5.

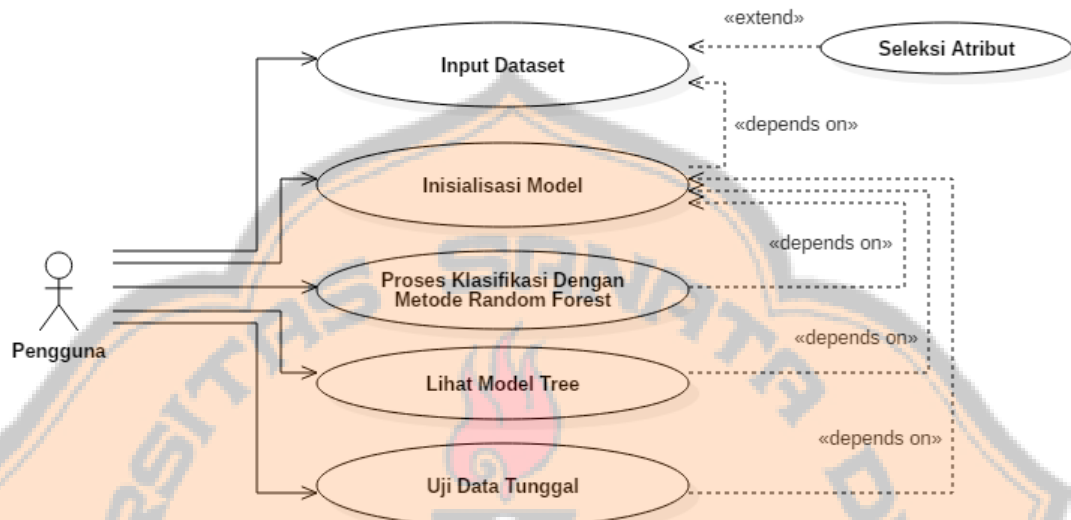
```

└── solo_KillDeathRatio: Root
    ├── (<2.03) Node: solo_KillDeathRatio
    │   ├── (<1.53) Node: solo_WinRatio leaf: 1.0
    │   └── (>=1.53) Node: solo_WinRatio leaf: 0.0
    └── (>=2.03) Node: solo_HeadshotKillsPg
        ├── (<0.59) Node: solo_WinRatio leaf: 1.0
        └── (>=0.59) Node: solo_WinRatio leaf: 2.0
  
```

Gambar 4. 5 Contoh Hasil Visualisasi *Tree*

4.2.2. Diagram *Use Case*

Diagram *Use Case* untuk sistem ini digambarkan pada gambar 4.6 berikut ini:



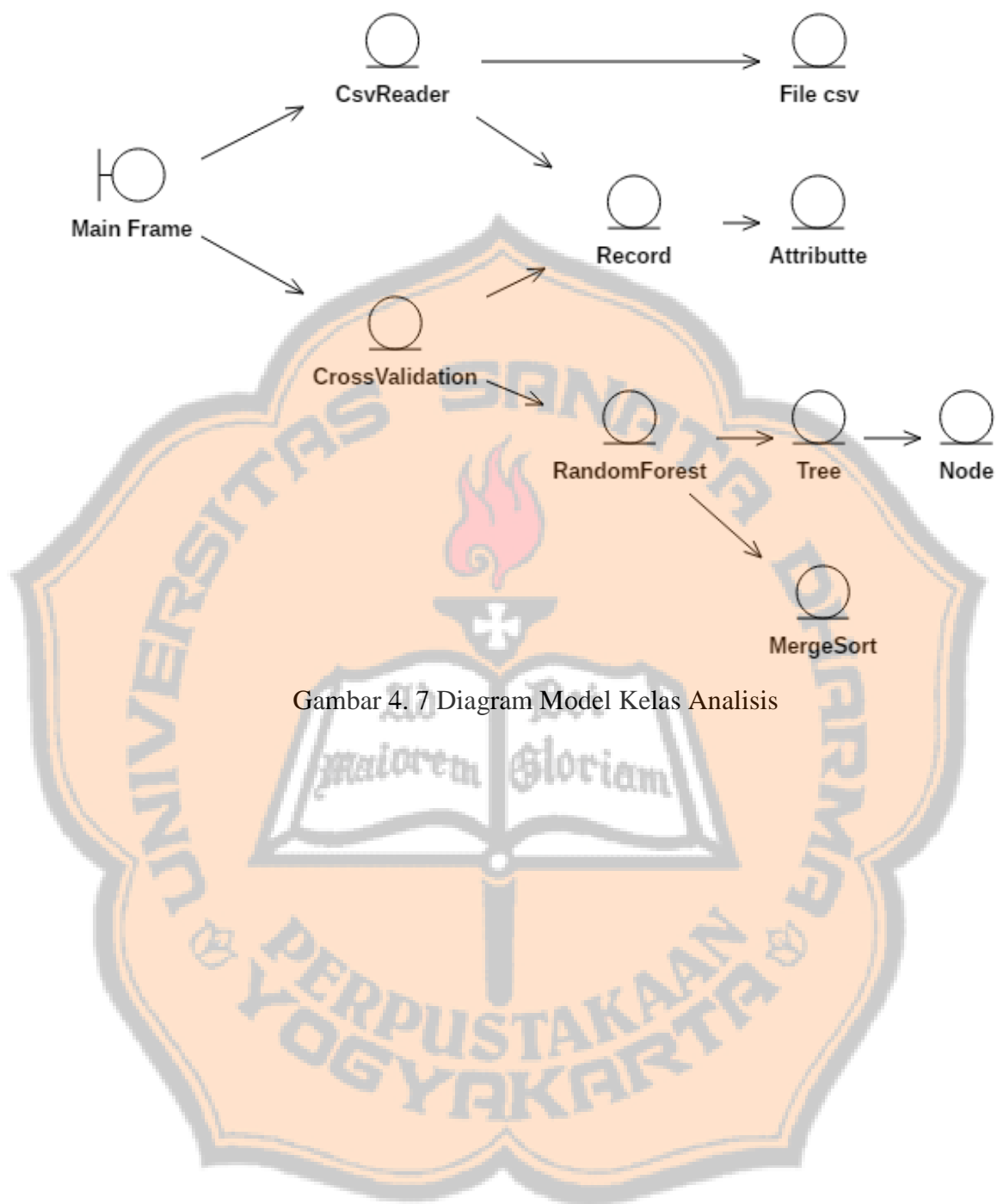
Gambar 4. 6 Diagram *Use Case*

4.2.3. Narasi *Use Case*

Dari diagram *use case* diatas, deskripsi dari setiap *use case* dijelaskan pada narasi *use case* yang terdapat pada lampiran I.

4.2.4. Diagram Kelas Analisis

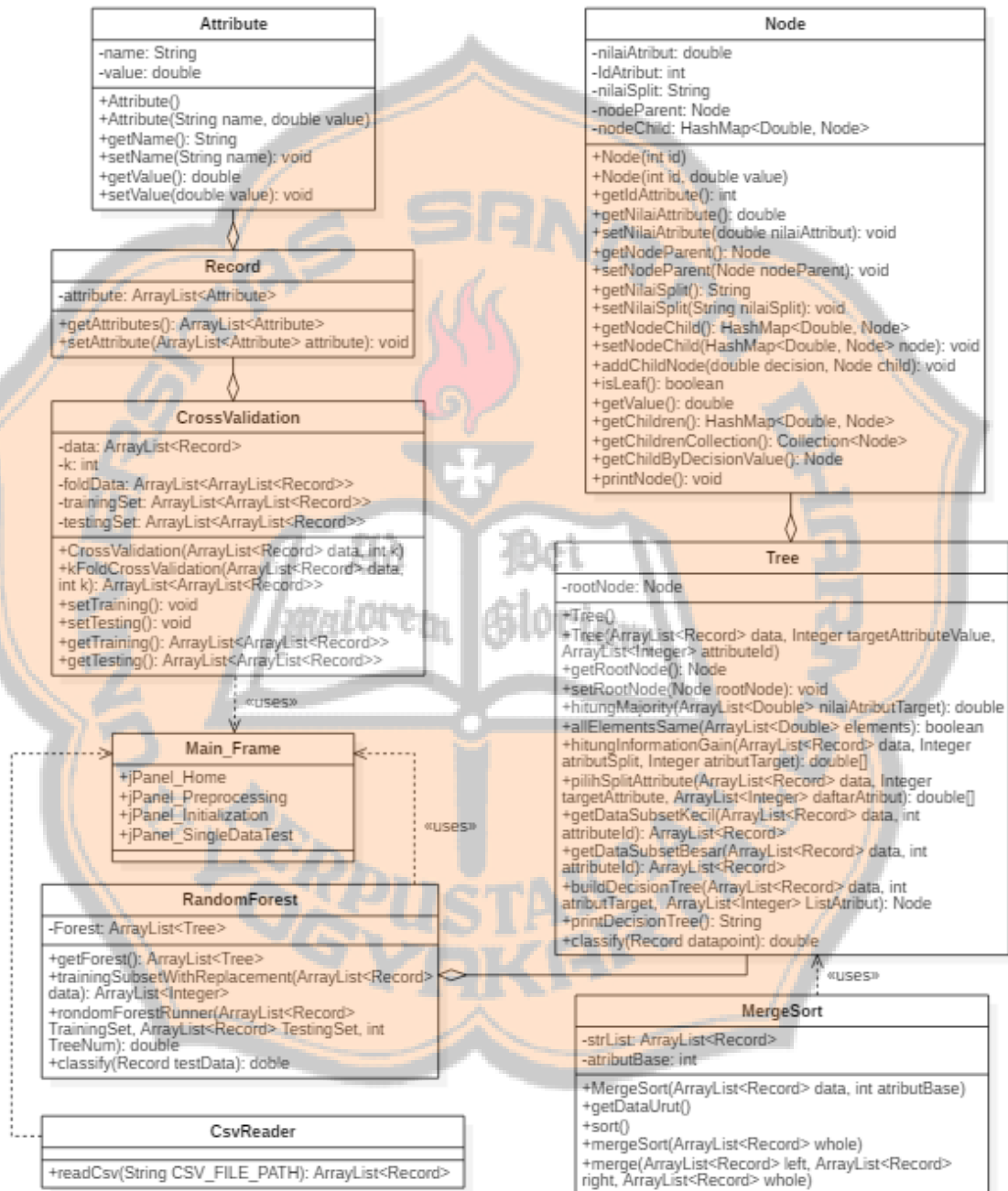
Gambar 4.7 berikut ini adalah diagram kelas analisis yang menggambarkan keterkaitan antar halaman *interface* dengan obyek kelas entitas:



Gambar 4. 7 Diagram Model Kelas Analisis

4.2.5. Diagram Kelas UML

Berdasarkan desain model kelas analisis pada gambar 4.7, maka diperoleh desain kelas UML yang ditunjukkan pada gambar 4.8.



Gambar 4. 8 Diagram Kelas UML

4.2.6. Perancangan Struktur Data

Dalam pembangunan sistem ini penulis menerapkan struktur data *arraylist* dan *hashmap*.

4.2.6.1. Arraylist

Dalam sistem ini, struktur data *arraylist* digunakan sebagai kontainer penyimpanan kumpulan data *record*. Hal tersebut dikarenakan struktur *arraylist* yang dinamis yang memungkinkan kita menambah atau mengurangi panjang kontainer tersebut agar dapat memanfaatkan *space* memori secara efektif. Selain itu *arraylist* juga digunakan dalam menyimpan objek *tree* dari setiap model *random forest* yang dibangun.

4.2.6.2. Hashmap

Dalam pembentukan *tree* dibutuhkan objek node dimana setiap node pada *tree* ada yang memiliki node child (node anak) kecuali jika node tersebut adalah node leaf (node ujung). Dalam melakukan penyimpanan objek node child sistem ini menggunakan struktur data *hashmap* yang dapat menyimpan sejumlah objek (node child) secara dinamis dengan key tertentu.

4.2.7. Algoritma per Metode

4.2.7.1. Kelas Model RandomForest

a) Metode randomForestRunner

Input :

1. ArrayList<Record> **TrainingSet**
2. ArrayList<Record> **TestingSet**
3. int **TreeNum**

Proses :

1. Lakukan perulangan sebanyak **TreeNum**
 - a. Buat training **subset** dari setiap array **TrainingSet** dengan menggunakan metode `trainingSubsetWithReplacement`
 - b. Bangun **tree** untuk dataset **subset**
 - c. Tambahkan **tree** ke atribut array **Forest** bertipe ArrayList<Tree>
2. Lakukan klasifikasi data dari array **Testingset** dengan menggunakan metode **classify**
3. Hitung persentase benar dari hasil klasifikasi, simpan hasil kedalam variabel **persentageCorrect** bertipe double
4. *Return* **persentageCorrect**

Output :

1. double **persentageCorrect**

b) **Metode classify**

Input :

1. Record **testData**

Proses :

1. Inisialisasi variabel **poll** bertipe
HashMap<Double, Integer>
2. Lakukan perulangan sebanyak jumlah Array
Forest dengan **i** dimulai dari 0 (langkah voting)
 - a. Masukan objek **Forest** ke **i** pada objek
Tree bernama **T**
 - b. Lakukan klasifikasi record **testData**
pada tree **T**, menggunakan metode
classify pada kelas **Tree**
 - c. Simpan hasil klasifikasi pada variabel
vote bertipe double
 - d. Jika Hashmap **poll** belum memiliki nilai
dengan key=**vote** maka masukan nilai
baru ke hasmap poll dengan key=**vote**
dan value=1
 - e. Jika sebaliknya, Hashmap **poll** sudah
memiliki nilai dengan key=**vote** maka
tambahkan value dengan 1 pada map
yang memiliki key=**vote**
3. Inisialisasi variabel **maxVote** = -1 bertipe int
4. Inisialisasi variabel **voteResult** = -1.0 bertipe
double
5. Lakukan perulangan sebanyak key yang
dimiliki hashmap **poll** menggunakan foreach
dengan element **labelKelas** bertipe Double
 - a. Inisialisasi variabel **vote** bertipe int

- b. Isi nilai **vote** dengan value dari map yang memiliki key **labelKelas**
- c. Jika nilai **vote** lebih dari **maxVote**, maka jadikan nilai **labelKelas** menjadi nilai **voteResult** dan jadikan juga nilai **vote** menjadi nilai **maxVote**

6. Return **voteResult**

Output :

- 1. double **voteResult**

c) **Metode trainingSubsetWithReplacement**

Input :

- 1. ArrayList<Record> **data**

Proses :

- 1. Buat objek **subset** bertipe arraylist int
- 2. Lakukan sampling random dengan pengembalian dari array **data** dengan hanya mengambil indeks dari array **data** tersebut
- 3. Masukkan hasil sampling ke arraylist **subset**
- 4. Jadikan **subset** sebagai return

Output :

- 1. ArrayList<Integer> **subset**

4.2.7.2. Kelas CrossValidation

a) Metode kFoldCrossValidation

Input :

1. ArrayList<Record> **data**

Proses :

1. Inisialisasi arraylist **FoldData** bertipe arrayList Record
2. Tentukan jumlah arraylist **FoldData** dengan membagi panjang array **data** dengan **fold**
3. Bagi array **data** sejumlah **fold**, masukan setiap bagian kedalam array **FoldData**
5. Return **FoldData**

Output :

1. ArrayList<ArrayList<Record>> **FoldData**

4.2.7.3. Kelas Tree

a) Metode buildDecisionTree

Input :

1. ArrayList<Record> **data**
2. int **atributTarget**
3. ArrayList<Integer> **ListAtribut**

Proses :

1. Hitung nilai majority setiap kelas target masukan ke variabel **nilaiMajority**
2. Jika ukuran **data** atau **ListAtribut** nol
 - a. Return objek Node dengan id = **atributTarget**, value = **nilaiMajority**, namaAttribute = nama **atributTarget**
3. Jika semua elemen dari **nilaiAtributTarget** sama

- a. Return objek Node dengan id = **atributTarget**, value = nilaiMajority, namaAttribute = nama **atributTarget**
4. Selain itu
 - a. Cari atribut terbaik untuk split data
 - b. Jadikan atribut terbaik sebagai **nodeBestAttribute** dengan tipe Node
 - c. Urutkan array **data** menggunakan objek **MergeSort**
 - d. Simpan **subsetBesar** dan **subsetKecil** dari hasil split nilai **nodeBestAttribute**
 - e. Buat subtree untuk masing-masing **subsetBesar** dan **subsetKecil**
 - f. Jadikan subtree **subsetBesar** sebagai child dari **nodeBestAttribute**, dengan key map=0
 - g. Jadikan subtree **subsetKecil** sebagai child dari **nodeBestAttribute**, dengan key map=1
 - h. Return **nodeBestAttribute**

Output :

1. Node **nodeBestAttribute**

b) Metode classify

Input :

1. Record **dataPoint**

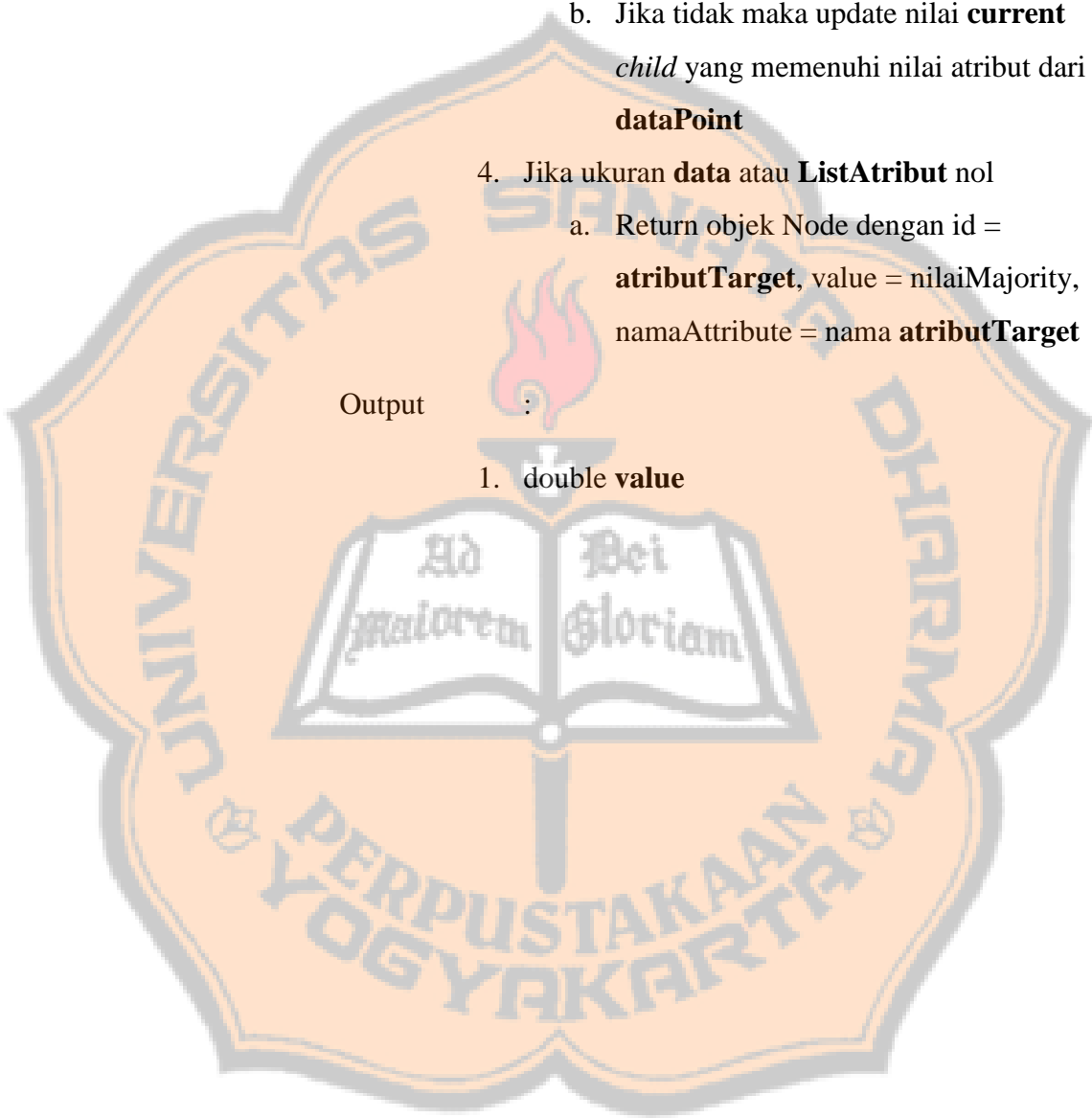
Proses :

1. Inisialisasi variabel **current** bertipe Node
2. Isi nilai **current** dengan atribut **rootNode**

3. Lakukan perulangan selama nilai **current** tidak null:
 - a. Jika **current** adalah leaf maka return **value** dari **current**
 - b. Jika tidak maka update nilai **current** *child* yang memenuhi nilai atribut dari **dataPoint**
4. Jika ukuran **data** atau **ListAtribut** nol
 - a. Return objek Node dengan id = **atributTarget**, value = nilaiMajority, namaAttribute = nama **atributTarget**

Output :

1. double **value**



4.2.8. Perancangan Antarmuka

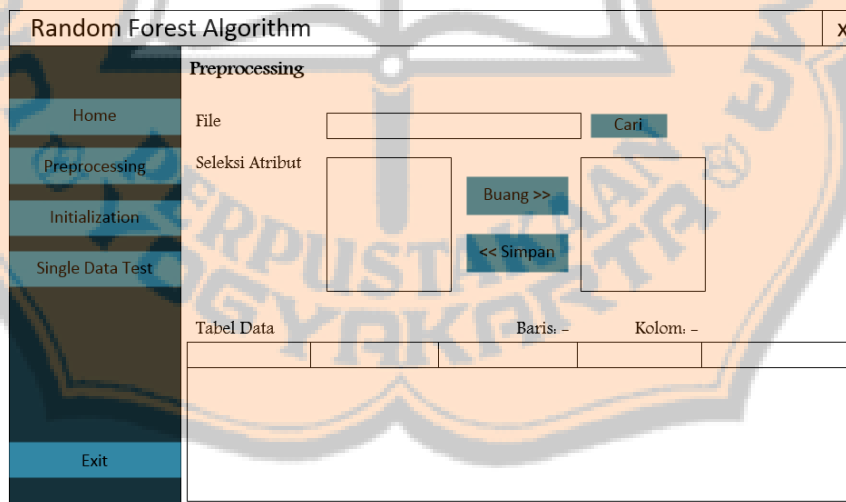
Gambar 4.9 sampai gambar 4.12 menunjukkan perancangan antarmuka sistem.

4.2.8.1. Halaman Awal



Gambar 4. 9 Desain Halaman Awal

4.2.8.2. Halaman Preprocessing



Gambar 4. 10 Desain Halaman Preprocessing

4.2.8.3. Halaman Initialization

Random Forest Algorithm		x
	Initialization	
Home	Jumlah Tree	<input type="text"/>
Preprocessing	Jumlah K-Fold	<input type="text"/>
Initialization	Run Algorithm	
Single Data Test	Akurasi	-
	Model Tree	<input type="text"/> ▼ <input type="text"/> ▼
	Tampilkan Dataset	
Exit		

Gambar 4. 11 Desain Halaman Initialization

4.2.8.4. Halaman Single Data Test

Random Forest Algorithm		x
	Single Data Test	
Home	Input Nilai Atribut	
Preprocessing		
Initialization		
Single Data Test		
Exit	Enter	
	Hasil Klasifikasi :	-

Gambar 4. 12 Desain Halaman Single Data Test

BAB V

IMPLEMENTASI DAN ANALISIS HASIL

5.1. Implementasi Perangkat Lunak

5.1.1. Implementasi Kelas *Model*

Tabel 5. 1 Implementasi kelas *model*

No	Nama Kelas	Nama File Fisik	Nama File Executable
1	Attribute	Attribute.java	Attribute.class
2	Record	Record.java	Record.class
3	TableModelData	TableModelData.java	TableModelData.class
4	Node	Node.java	Node.class
5	Tree	Tree.java	Tree.class
6	RandomForest	RandomForest.java	RandomForest.class

5.1.1. Implementasi Kelas *Control*

Tabel 5. 2 Implementasi kelas *control*

No	Use Case	Nama File Fisik	Nama File Executable
1	CrossValidation	CrossValidation.java	CrossValidation.class
2	Proses Klasifikasi Dengan Metode <i>Random Forest</i>	CsvReader.java	CsvReader.class
3	Proses Klasifikasi Dengan Metode <i>Random Forest</i>	MergeSort.java	MergeSort.class

5.1.2. Implementasi Kelas *View*

Tabel 5. 3 Implementasi kelas *view*

No	Use Case	Nama <i>File</i> Fisik	Nama <i>File Executable</i>
1	Input <i>Dataset</i>	MainFrame.java	MainFrame.class
2	Seleksi Atribut	MainFrame.java	MainFrame.class
3	Inisialisasi Model	MainFrame.java	MainFrame.class
4	Proses Klasifikasi Dengan Metode <i>Random Forest</i>	MainFrame.java	MainFrame.class
5	Lihat Model <i>Tree</i>	MainFrame.java	MainFrame.class
6	Uji Data Tunggal	MainFrame.java	MainFrame.class

5.2. Analisis Hasil

5.2.1. Pengujian Perangkat Lunak

5.2.1.1. Prosedur Pengujian dan Kasus Uji

Penulis melakukan prosedur pengujian beserta kasus uji yang dapat dilihat pada lampiran II. Prosedur beserta hasil pengujian terdapat pada tabel 6.7.

5.2.1.2. Evaluasi Pengujian

Dari semua hasil pengujian yang terlampir pada lampiran II menunjukkan bahwa perangkat lunak dapat berjalan dengan baik sesuai dengan perancangan yang sudah dibuat. Hal tersebut didukung dari hasil pengujian fungsi-fungsi yang ada berjalan sesuai yang diharapkan. Selain itu perangkat lunak ini juga mampu mengatasi kesalahan-kesalahan yang dilakukan *user* dengan menampilkan pesan kesalahan. Hal tersebut sangat baik dikarenakan dengan adanya pesan-pesan kesalahan maka *user* mudah mengerti

kesalahan apa yang sudah dilakukan sehingga dapat menghindari kesalahan yang sama di masa yang akan datang.

5.2.2. Pengujian Perbandingan Hasil Hitung Manual dengan Hasil Perangkat Lunak

5.2.2.1. Penghitungan Perangkat Lunak

Pada Proses pengujian perhitungan menggunakan perangkat lunak dengan data statistik Pemain *Player Unknown Battleground* yang diambil secara acak. Proses pembangunan model *Random Forest* menggunakan perangkat lunak dengan memasukan *file csv* yang berisi 12 data yang diambil secara acak dari data statistik Pemain *Player Unknown Battleground*. Atribut yang digunakan adalah `solo_KillDeathRatio`, `solo_DamagePg`, `solo_HeadshotKillsPg`, `solo_MoveDistancePg` dan `solo_TimeSurvivedPg` seperti yang ditunjukkan pada gambar 5.1.

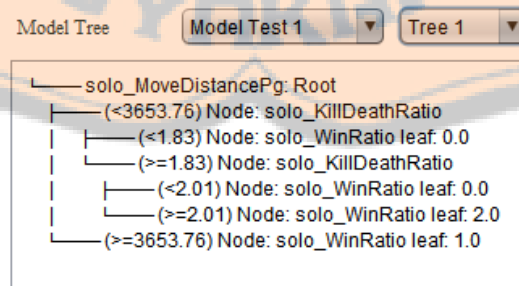
No	solo_Win...	solo_Kill...	solo_Da...	solo_Hea...	solo_Mov...	solo_Tim...
1	1.0	1.23	151.25	0.18	3653.76	1048.71
2	0.0	1.83	193.0	0.38	1358.2	920.77
3	2.0	2.01	225.11	0.44	3039.84	1004.96
4	2.0	2.1	217.86	0.5	4970.75	1353.75
5	0.0	1.02	124.03	0.14	2731.76	999.58
6	1.0	1.08	123.32	0.22	3701.09	1077.55
7	0.0	1.04	129.73	0.23	2737.54	970.17
8	1.0	1.05	129.18	0.25	2863.96	867.6
9	2.0	2.03	210.54	0.59	3986.74	1171.84
10	0.0	1.53	189.73	0.34	2170.23	819.47
11	1.0	2.14	241.63	0.58	2178.41	804.51
12	2.0	1.06	125.64	0.21	3312.12	1062.95

Gambar 5. 1 Preprocessing Penghitungan Perangkat Lunak

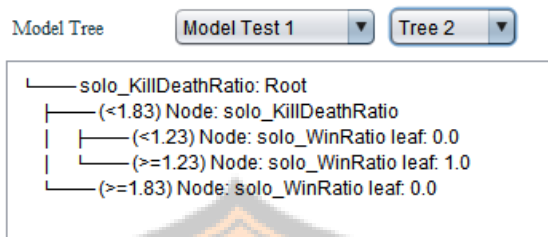
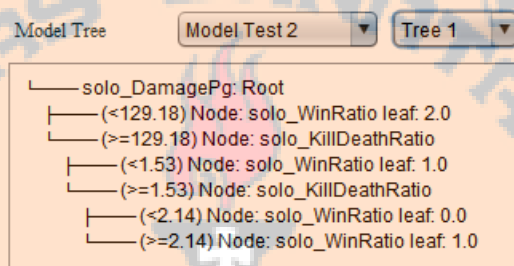
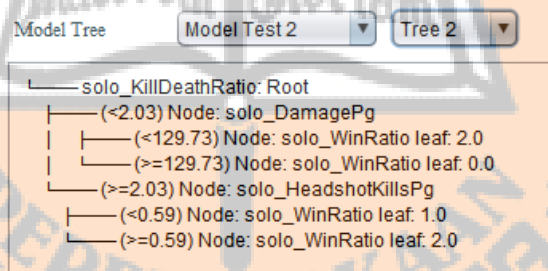
Inisialisasi pengujian menggunakan jumlah *fold* 2 dan jumlah *tree* 2. Dengan inisialisasi tersebut maka akan menghasilkan 2 model dengan 2 *tree* pada masing masing model tersebut. Akurasi yang didapatkan oleh sistem adalah sebesar 33,33% yang dapat dilihat pada gambar 5.2. Hasil pembentukan *tree* oleh sistem dapat dilihat pada gambar 5.3 sampai 5.6. Untuk hasil sampling yang dilakukan oleh sistem yang akan digunakan untuk membangun *tree* dapat dilihat pada gambar 5.7 sampai 5.10.



Gambar 5. 2 Hasil Akurasi Penghitungan Sistem



Gambar 5. 3 Hasil Pembentukan *Tree* 1 Pada Model 1 Oleh Sistem

Gambar 5. 4 Hasil Pembentukan *Tree 2* Pada Model 1 Oleh SistemGambar 5. 5 Hasil Pembentukan *Tree 1* Pada Model 2 Oleh SistemGambar 5. 6 Hasil Pembentukan *Tree 2* Pada Model 2 Oleh Sistem

No	solo_Win...	solo_Kill...	solo_Da...	solo_Hea...	solo_Mov...	solo_Tim...
1	0.0	1.83	193.0	0.38	1358.2	920.77
2	0.0	1.02	124.03	0.14	2731.76	999.58
3	2.0	2.01	225.11	0.44	3039.84	1004.96
4	1.0	1.23	151.25	0.18	3653.76	1048.71
5	1.0	1.08	123.32	0.22	3701.09	1077.55
6	1.0	1.08	123.32	0.22	3701.09	1077.55

Gambar 5. 7 Hasil Sampling Dataset Untuk *Tree 1* Pada Model 1 Oleh Sistem

No	solo_Win...	solo_Kill...	solo_Da...	solo_Hea...	solo_Mov...	solo_Tim...
1	0.0	1.02	124.03	0.14	2731.76	999.58
2	1.0	1.23	151.25	0.18	3653.76	1048.71
3	1.0	1.23	151.25	0.18	3653.76	1048.71
4	0.0	1.83	193.0	0.38	1358.2	920.77
5	0.0	1.83	193.0	0.38	1358.2	920.77
6	2.0	2.1	217.86	0.5	4970.75	1353.75

Gambar 5. 8 Hasil Sampling Dataset Untuk *Tree 2* Pada Model 1 Oleh Sistem

No	solo_Win...	solo_Kill...	solo_Da...	solo_Hea...	solo_Mov...	solo_Tim...
1	2.0	1.06	125.64	0.21	3312.12	1062.95
2	2.0	1.06	125.64	0.21	3312.12	1062.95
3	1.0	1.05	129.18	0.25	2863.96	867.6
4	1.0	1.05	129.18	0.25	2863.96	867.6
5	0.0	1.53	189.73	0.34	2170.23	819.47
6	1.0	2.14	241.63	0.58	2178.41	804.51

Gambar 5. 9 Hasil Sampling Dataset Untuk *Tree 1* Pada Model 2 Oleh Sistem

No	solo_Win...	solo_Kill...	solo_Da...	solo_Hea...	solo_Mov...	solo_Tim...
1	0.0	1.04	129.73	0.23	2737.54	970.17
2	0.0	1.04	129.73	0.23	2737.54	970.17
3	2.0	1.06	125.64	0.21	3312.12	1062.95
4	2.0	2.03	210.54	0.59	3986.74	1171.84
5	2.0	2.03	210.54	0.59	3986.74	1171.84
6	1.0	2.14	241.63	0.58	2178.41	804.51

Gambar 5. 10 Hasil Sampling Dataset Untuk *Tree 2* Pada Model 2 Oleh Sistem

5.2.2.2. Penghitungan Manual

Pada proses pengujian penghitungan manual *Random Forest* menggunakan data yang sama dengan pengujian penghitungan perangkat lunak yaitu data statistik Pemain *Player Unknown Battleground*. Proses penghitungan manual menggunakan *Microsoft Excel*.

Untuk penghitungan manual pembangunan model *Random Forest* menggunakan data yang berhasil diambil secara acak oleh perangkat lunak dengan mengakses menu *show data* pada halaman *initialization*. Dataset tersebut dapat dilihat pada gambar 5.7 sampai 5.10. Pada pengujian perangkat lunak menggunakan jumlah *fold* 2 dan *tree* sejumlah 2, maka dalam perhitungan manual akan menggunakan 4 *dataset* untuk membuat 4 *tree* dengan 2 *tree* untuk setiap model *random forest* pada kedua model. Proses perhitungan manual beserta dengan hasilnya dapat dilihat pada lampiran III.

5.2.2.3. Evaluasi Pengujian

Hasil perhitungan manual dengan perhitungan perangkat lunak tidak memiliki perbedaan, baik itu dilihat dari model *tree* yang sudah dibangun maupun hasil akurasi final yang didapatkan. Perbandingan bentuk *tree* yang dihasilkan oleh sistem dan penghitungan manual dapat dilihat pada tabel 5.4. Untuk hasil akurasi final yang diperoleh dari penghitungan sistem sebesar 25% dan penghitungan manual juga sebesar 25%. Dengan begitu dapat disimpulkan bahwa perangkat lunak sudah berjalan dengan sangat baik dan sesuai dengan yang diharapkan.

Tabel 5. 4 Perbandingan *Tree* Hasil Sistem dan Perhitungan Manual

Model	Tree	Hasil Sistem	Hasil Penghitungan Manual
Model 1	Tree 1	<p>Model Tree Model Test 1 Tree 1</p> <pre> └─ solo_MoveDistancePg: Root ├─ (<3653,76) Node: solo_KillDeathRatio │ ├─ (<1.83) Node: solo_WinRatio leaf: 0.0 │ └─ (>=1.83) Node: solo_KillDeathRatio │ ├─ (<2.01) Node: solo_WinRatio leaf: 0.0 │ └─ (>=2.01) Node: solo_WinRatio leaf: 2.0 └─ (>=3653,76) Node: solo_WinRatio leaf: 1.0 </pre>	<pre> └─ solo_MoveDistancePg: Root ├─ (<3653,76) Node: solo_KillDeathRatio │ ├─ (<1.83) Node: solo_WinRatio leaf: 0 │ └─ (>=1.83) Node: solo_KillDeathRatio │ ├─ (<2.01) Node: solo_WinRatio leaf: 0 │ └─ (>=2.01) Node: solo_WinRatio leaf: 2 └─ (>=3653,76) Node: solo_WinRatio leaf: 1 </pre>
	Tree 2	<p>Model Tree Model Test 1 Tree 2</p> <pre> └─ solo_KillDeathRatio: Root ├─ (<1.83) Node: solo_KillDeathRatio │ ├─ (<1.23) Node: solo_WinRatio leaf: 0.0 │ └─ (>=1.23) Node: solo_WinRatio leaf: 1.0 └─ (>=1.83) Node: solo_WinRatio leaf: 0.0 </pre>	<pre> └─ solo_KillDeathRatio: Root ├─ (<1.83) Node: solo_KillDeathRatio │ ├─ (<1,23) Node: solo_WinRatio leaf: 0 │ └─ (>=1,23) Node: solo_WinRatio leaf: 1 └─ (>=1,83) Node: solo_WinRatio leaf: 0 </pre>
Model 2	Tree 1	<p>Model Tree Model Test 2 Tree 1</p> <pre> └─ solo_DamagePg: Root ├─ (<129,18) Node: solo_WinRatio leaf: 2.0 └─ (>=129,18) Node: solo_KillDeathRatio ├─ (<1.53) Node: solo_WinRatio leaf: 1.0 └─ (>=1.53) Node: solo_KillDeathRatio ├─ (<2,14) Node: solo_WinRatio leaf: 0.0 └─ (>=2,14) Node: solo_WinRatio leaf: 1.0 </pre>	<pre> └─ solo_DamagePg: Root ├─ (<129,18) Node: solo_WinRatio leaf: 2 └─ (>=129,18) Node: solo_KillDeathRatio ├─ (<1.53) Node: solo_WinRatio leaf: 2 └─ (>=1.53) Node: solo_KillDeathRatio ├─ (<2,14) Node: solo_WinRatio leaf: 0 └─ (>=2,14) Node: solo_WinRatio leaf: 1 </pre>

	Tree 2	<div> Model Tree Model Test 2 Tree 2 </div> <div> <pre> └─ solo_KillDeathRatio: Root └─ (<2.03) Node: solo_DamagePg └─ (<129.73) Node: solo_WinRatio leaf: 2.0 └─ (>=129.73) Node: solo_WinRatio leaf: 0.0 └─ (>=2.03) Node: solo_HeadshotKillsPg └─ (<0.59) Node: solo_WinRatio leaf: 1.0 └─ (>=0.59) Node: solo_WinRatio leaf: 2.0 </pre> </div>	<pre> └─ solo_KillDeathRatio: Root └─ (<2,03) Node: solo_DamagePg └─ (<129,18) Node: solo_WinRatio leaf: 0 └─ (>=129,18) Node: solo_WinRatio leaf: 2 └─ (>=2,03) Node: solo_HeadshotKillsPg └─ (<0,59) Node: solo_WinRatio leaf: 1 └─ (>=0,59) Node: solo_WinRatio leaf: 2 </pre>
--	--------	--	---

5.2.3. Pengujian Perangkat Lunak Dengan Menggunakan *Dataset*

Pada pengujian perangkat lunak menggunakan *dataset* akan membandingkan hasil akurasi yang dihasilkan dari *dataset* berikut ini:

1. *Dataset* statistik Pemain *Player Unknown Battleground* dengan *outlier* dan data tidak ternormalisasi.
2. *Dataset* statistik Pemain *Player Unknown Battleground* dengan *outlier* dan data ternormalisasi.
3. *Dataset* statistik Pemain *Player Unknown Battleground* tanpa *outlier* dan data tidak ternormalisasi.
4. *Dataset* statistik Pemain *Player Unknown Battleground* tanpa *outlier* dan data ternormalisasi.

Pada pengujian ini akan dilakukan pengujian klasifikasi data statistik Pemain *Player Unknown Battleground* dengan jumlah *fold* 3 dan kombinasi jumlah *tree* yang berbeda-beda dimulai dari 10 sampai 90 dengan interval 10. Dari pengujian ini akan dilihat hasil akurasi dari masing-masing pengujian.

5.2.3.1. Pengujian *Dataset* statistik Pemain *Player Unknown Battleground* dengan *outlier* dan data tidak ternormalisasi.

Dataset statistik Pemain *Player Unknown Battleground* dengan *outlier* dan data tidak ternormalisasi memiliki 87.898 baris data dan 30 atribut. Tabel pengujian *dataset* statistik Pemain *Player Unknown Battleground* tanpa *outlier* dapat dilihat di tabel 5.5.

Tabel 5. 5 Pengujian dataset statistik Pemain *Player Unknown Battleground* dengan outlier dan data tidak ternormalisasi

No.	Jumlah <i>Tree</i>	Akurasi (%)
1	10	86,11
2	20	87,14
3	30	87,55
4	40	87,85
5	50	87,85
6	60	87,86
7	70	88,19
8	80	87,81

5.2.3.2. Pengujian *Dataset* statistik Pemain *Player Unknown Battleground* dengan outlier dan data ternormalisasi.

Dataset statistik Pemain *Player Unknown Battleground* dengan outlier dan data ternormalisasi memiliki 87.898 baris data dan 30 atribut. Tabel pengujian *Dataset* statistik Pemain *Player Unknown Battleground* dengan outlier dan data ternormalisasi dapat dilihat di tabel 5.6.

Tabel 5. 6 Pengujian *dataset* statistik Pemain *Player Unknown Battleground* dengan outlier dan data ternormalisasi

No.	Jumlah <i>Tree</i>	Akurasi (%)
1	10	84,14
2	20	85,25
3	30	85,60
4	40	85,94
5	50	85,70
6	60	86,08
7	70	86,00
8	80	86,08

5.2.3.3. Pengujian *Dataset* statistik Pemain *Player Unknown Battleground* tanpa outlier dan data tidak ternormalisasi.

Dataset statistik Pemain *Player Unknown Battleground* tanpa outlier dan data tidak ternormalisasi memiliki 58.366 baris data dan 30 atribut. Tabel pengujian *dataset* statistik Pemain *Player Unknown Battleground* tanpa outlier dan data tidak ternormalisasi dapat dilihat di tabel 5.7.

Tabel 5. 7 Pengujian *dataset* statistik Pemain *Player Unknown Battleground* tanpa outlier dan data tidak ternormalisasi

No.	Jumlah <i>Tree</i>	Akurasi (%)
1	10	74,55
2	20	76,32
3	30	77,09
4	40	77,68
5	50	77,62
6	60	77,88
7	70	77,90
8	80	78,19

5.2.3.4. Pengujian *Dataset* statistik Pemain *Player Unknown Battleground* tanpa *outlier* dan data ternormalisasi.

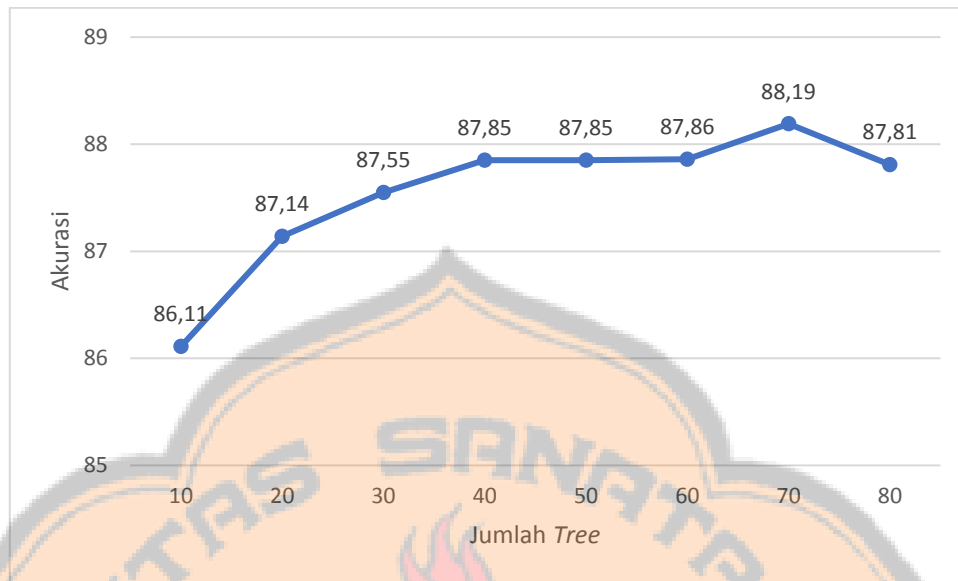
Dataset statistik Pemain *Player Unknown Battleground* tanpa *outlier* dan data ternormalisasi memiliki 58.366 baris data dan 30 atribut. Tabel pengujian *dataset* statistik Pemain *Player Unknown Battleground* dengan *outlier* dan data ternormalisasi dapat dilihat di tabel 5.8.

Tabel 5. 8 Pengujian *dataset* statistik Pemain *Player Unknown Battleground* dengan *outlier* dan data ternormalisasi

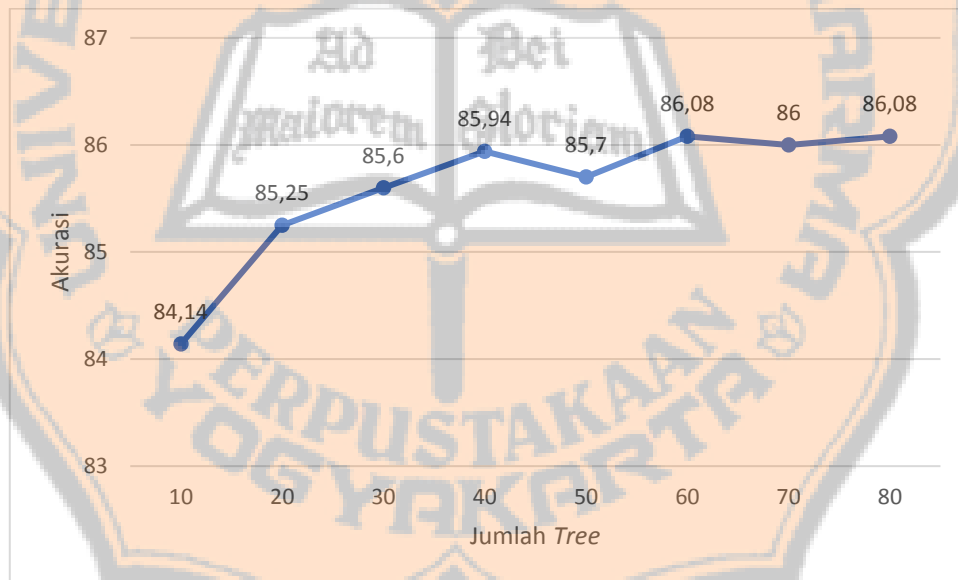
No.	Jumlah <i>Tree</i>	Akurasi (%)
1	10	74,77
2	20	76,53
3	30	76,86
4	40	77,67
5	50	77,67
6	60	77,78
7	70	77,94
8	80	78,18

5.2.3.5. Evaluasi Hasil Pengujian

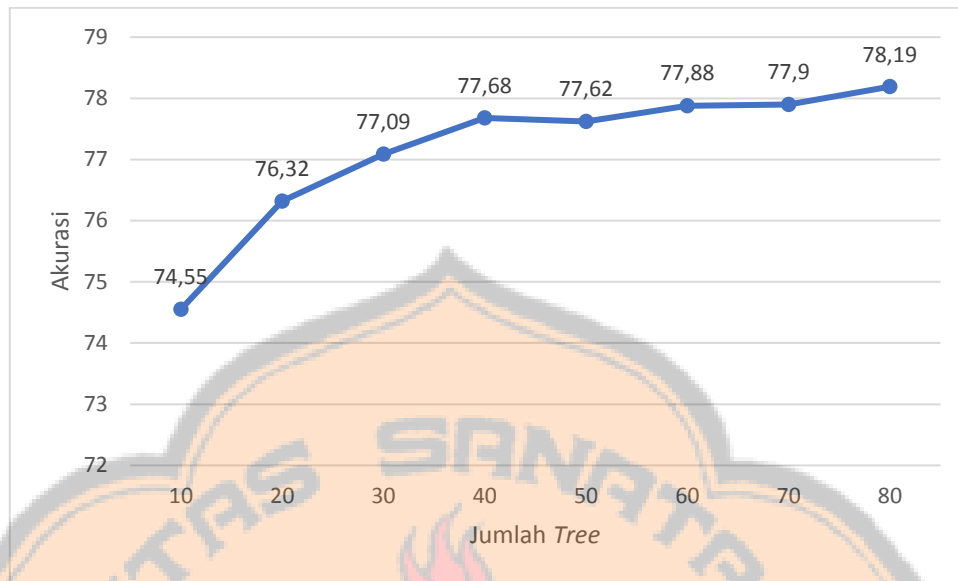
Dari hasil pengujian ke empat dataset tersebut terdapat kesamaan pola pada pengujian keempat *dataset* tersebut yaitu akurasi yang cenderung bertambah ketika jumlah *tree* semakin besar. Pola tersebut dapat mudah di lihat pada grafik yang digambarkan pada gambar 5.11 sampai 5.14. Hal tersebut menunjukan bahwa semakin banyak jumlah *tree* yang digunakan dalam membangun model maka akurasi cenderung meningkat.



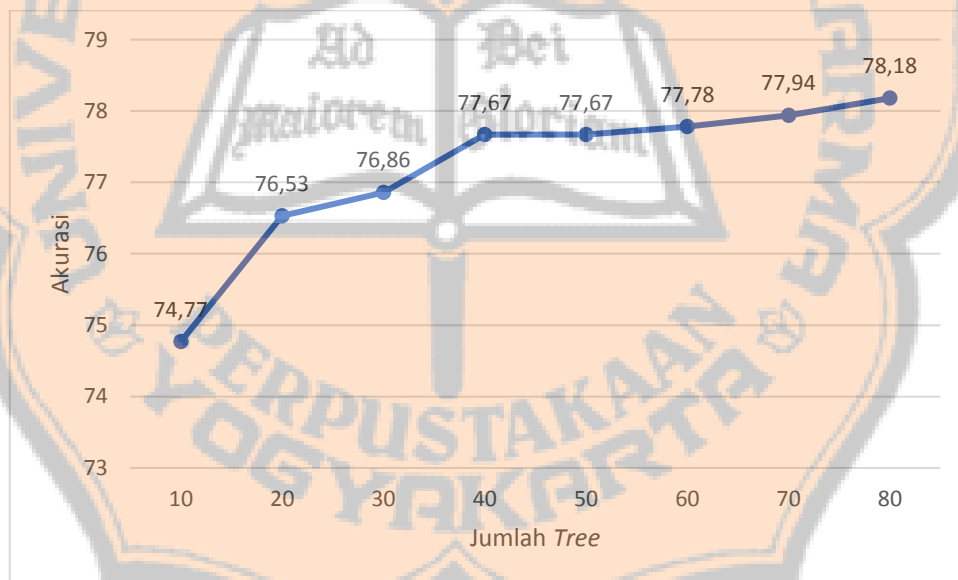
Gambar 5. 11 Hasil pengujian dataset statistik pemain *Player Unknown Battleground* dengan outlier dan data tidak ternormalisasi



Gambar 5. 12 Hasil pengujian dataset statistik pemain *Player Unknown Battleground* dengan outlier dan data ternormalisasi



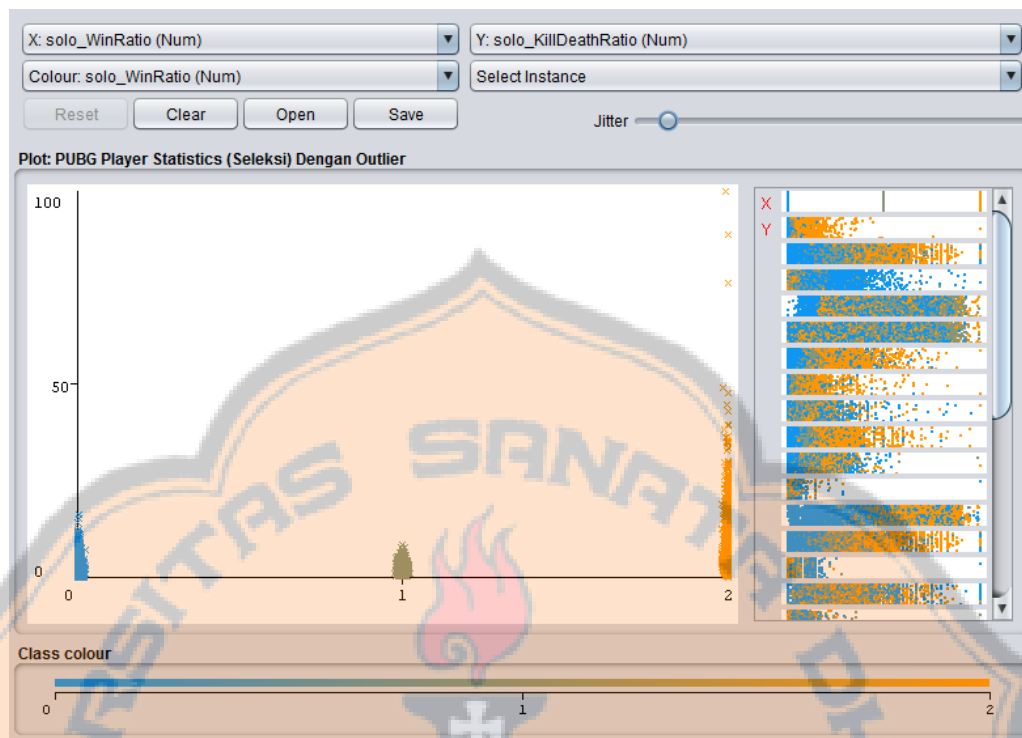
Gambar 5. 13 Hasil pengujian dataset statistik pemain *Player Unknown Battleground* tanpa outlier dan data tidak ternormalisasi



Gambar 5. 14 Hasil pengujian dataset statistik pemain *Player Unknown Battleground* tanpa outlier dan data ternormalisasi

Dari hasil pengujian keempat dataset tersebut didapatkan akurasi tertinggi yaitu 88,19%. Akurasi tersebut didapat dari hasil klasifikasi *dataset* statistik Pemain *Player Unknown Battleground* dengan outlier dan data tidak ternormalisasi dengan jumlah *tree* sebanyak 70. Jika dilihat model yang dibuat untuk mendapatkan hasil tertinggi tersebut, atribut yang sering digunakan sebagai *root node* maupun node lain adalah atribut *solo_KillDeathRatio*. Bahkan atribut tersebut digunakan sebagai *root node* di seluruh *tree* (sebanyak 70 *tree*). Artinya atribut tersebut sangat berperan besar dalam memprediksi atau mengklasifikasi *win ratio*.

Dengan memanfaatkan tools Weka, penulis mencoba untuk melihat bagaimana korelasi atribut tersebut dengan *win ratio*. Jika dilihat dari scatter plot gambar 5.15, ada cukup banyak nilai *solo_KillDeathRatio* yang tinggi yang menunjukkan *win ratio* tinggi (kluster 2) artinya jika nilai split diletakan pada tengah atribut *solo_KillDeathRatio* maka ada cukup banyak data yang memiliki *win ratio* tinggi (kluster 2) yang terdapat pada salah satu subset. Hal tersebutlah yang menyebabkan atribut ini selalu memiliki nilai *information gain* tinggi dari pada atribut lain.



Gambar 5. 15 Korelasi atribut solo_KillDeathRatio dengan win ratio

BAB VI

PENUTUP

6.1. Kesimpulan

Hasil penelitian penerapan metode *random forest* untuk prediksi *win ratio* pemain *player unknown battleground* ini menghasilkan kesimpulan sebagai berikut:

- 1) Metode *random forest* dapat digunakan untuk melakukan prediksi/klasifikasi *win ratio* pemain *player unknown battleground* dengan baik. Semakin banyak jumlah *tree* yang digunakan maka semakin baik pula akurasi yang didapat.
- 2) Atribut yang paling berpengaruh dalam menentukan klasifikasi *win ratio* pemain *player unknown battleground* adalah *solo_KillDeathRatio*.
- 3) Akurasi prediksi terbaik yang dihasilkan oleh metode *Random Forest* terhadap data statistik pemain game *Player Unknown Battleground* adalah 88,19%.

6.2. Saran

Penelitian penerapan metode *random forest* untuk prediksi *win ratio* pemain *player unknown battleground* ini memberikan saran untuk pengembangan penelitian yang akan datang, yaitu:

- 1) Perangkat lunak dapat menyimpan atau mengeksport model yang berhasil dibangun oleh perangkat lunak
- 2) Perangkat lunak dapat menampilkan semua rule yang memiliki *leaf* suatu label.
- 3) Perangkat lunak dapat melakukan transformasi/normalisasi data dengan sendiri
- 4) Penelitian menggunakan *dataset* yang berbeda

DAFTAR PUSTAKA

- Anonim. 2013. Ini Plus Minusnya Bermain Game Digital Bagi Anak. di <https://health.detik.com/ulasan-khas/d-2418829/ini-plus-minusnya-bermain-game-digital-bagi-anak> (di akses November)
- Breiman Leo.2001.*Machine Learning*.Berkeley:University of California
- Han, Jiawei.2012. *Data Mining Concepts and Techniques Third Edition*. USA:Elsevier
- Jannah, Selfie Miftahul.2018. *Bukan Cuma Main Game, Esport Mulai Jadi Industri Masa Depan* di <https://finance.detik.com/berita-ekonomi-bisnis/d-4316768/bukan-cuma-main-game-esport-mulai-jadi-industri-masa-depan> (di akses November)
- Jati, Anggoro Suryo.2018.*UniPin Bikin Kompetisi eSports Berhadiah Rp 1,4 Miliar* di <https://inet.detik.com/games-news/d-4162385/unipin-bikin-kompetisi-esports-berhadiah-rp-14-miliar> (di akses November)
- Nidhomiddin & Otok.2015.*RANDOM FOREST DAN MULTIVARIATE ADAPTIVE REGRESSION SPLINE (MARS) BINARY RESPONSE UNTUK KLASIFIKASI PENDERITA HIV/AIDS DI SURABAYA*.Bandung:Institut Teknologi Sepuluh November Surabaya.
- Nugroho.2017.*Sistem Klasifikasi Variabel Tingkat Penerimaan Konsumen Terhadap Mobil Menggunakan Metode Random Forest* di https://www.researchgate.net/publication/320413581_Sistem_Klasifikasi_Variabel_Tingkat_Penerimaan_Konsumen_Terhadap_Mobil_Menggunakan_Metode_Random_Forest (diakses November)
- Polamuri, Saimadhu. How Random Forest Algorithm Works In Machine Learning <https://dataaspirant.com/2017/05/22/random-forest-algorithm-machine-learning>
- Refaeilzadeh, Payam.2008.*Cross Validation* di <http://leidang.net/papers/ency-cross-validation.pdf> (diakses November)
- Restika, Ria.2018. *Apa Itu Esports?* di <https://esportsnesia.com/penting/apa-itu-esports/> (di akses November)





LAMPIRAN

LAMPIRAN I: NARASI *USE CASE*1. Narasi *Use Case* Input *Dataset*Tabel 6. 1 Narasi *Use Case* Input *Dataset*

Input <i>Dataset</i>		
Nama Use Case	Input <i>Dataset</i>	
ID Use Case	1	
Aktor	User	
Deskripsi	<i>Use case</i> ini merupakan proses memasukan <i>dataset</i> dari file .csv ke dalam sistem	
Kondisi Awal	User telah masuk ke dalam sistem dan berada di halaman preprocessing	
Kondisi Akhir	Data ditampilkan dalam tabel data pada halaman <i>preprocessing</i>	
Typical Course	Aksi Aktor	Reaksi Sistem
	1) Klik tombol “Cari” untuk memasukan <i>file</i> yang berekstensi .csv	
		2) Menampilkan dialog pencarian <i>file</i>
	3) Memilih <i>file</i> yang akan digunakan sebagai <i>dataset</i>	
		4) Menampilkan data dari <i>file</i> yang dipilih ke tabel data pada halaman <i>Preprocessing</i>
Alternate Course	-	-

2. Narasi *Use Case* Seleksi Atribut

Tabel 6. 2 Narasi Use Case Seleksi Atribut

Seleksi Atribut		
Nama Use Case	Seleksi Atribut	
ID Use Case	2	
Aktor	User	
Deskripsi	<i>Use case</i> ini merupakan proses pemilihan atribut dari data yang sudah dimasukan	
Kondisi Awal	User telah masuk ke dalam sistem dan berada di halaman preprocessing dan data yang sudah dimasukan tampil di halaman <i>preprocessing</i>	
Kondisi Akhir	Pemilihan atribut yang akan di klasifikasi selesai. Data yang dipilih ditampilkan di halaman <i>preprocessing</i>	
Typical Course	Aksi Aktor	Reaksi Sistem
	1) Memilih satu persatu atribut yang akan dibuang dari <i>dataset</i> lalu klik buang	
		2) Menampilkan <i>dataset</i> pada tabel data di halaman <i>Preprocessing</i> dengan menghilangkan atribut yang dipilih oleh user
Alternate Course	Aksi Aktor	Reaksi Sistem
	1) Memilih semua atribut yang akan dibuang dengan menahan tombol <i>Ctrl</i> lalu klik buang	

		2) Menampilkan <i>dataset</i> pada tabel data di halaman <i>Preprocessing</i> dengan menghilangkan atribut yang dipilih oleh user
--	--	---

3. Narasi *Use Case* Inisialisasi Model

Tabel 6. 3 Narasi Use Case Inisialisasi Model

Inisialisasi Model		
Nama Use Case	Inisialisasi Model	
ID Use Case	3	
Aktor	User	
Deskripsi	<i>Use case</i> ini merupakan proses melakukan inisialisasi model sebelum model di bentuk	
Kondisi Awal	<i>User</i> telah masuk ke dalam sistem dan berada pada halaman <i>initialization</i>	
Kondisi Akhir	Field jumlah <i>tree</i> dan jumlah K-Fold sudah terisi	
Typical Course	Aksi Aktor	Reaksi Sistem
	1) Mengisi field “Jumlah <i>Tree</i> ” dan “Jumlah K-Fold”	2) Menampilkan “Jumlah <i>Tree</i> ” dan “Jumlah K-Fold”
Alternate Course	-	-

4. Narasi *Use Case* Proses Klasifikasi Dengan Metode *Random Forest*Tabel 6. 4 Narasi *Use Case* Proses Klasifikasi Dengan Metode *Random Forest*

Proses Klasifikasi Dengan Metode <i>Random Forest</i>		
Nama Use Case	Proses Klasifikasi Dengan metode <i>Random Forest</i>	
ID Use Case	4	
Aktor	User	
Deskripsi	<i>Use case</i> ini merupakan proses klasifikasi dari data yang sudah dipilih dan sudah diseleksi	
Kondisi Awal	User sudah masuk dalam sistem dan berada di halaman <i>initialization</i>	
Kondisi Akhir	Sistem menampilkan hasil akurasi dan model <i>tree</i> yang berhasil dibangun	
Typical Course	Aksi Aktor	Reaksi Sistem
	1) Klik tombol “Run Algorithm”	
		2) Menampilkan hasil akurasi dan model <i>tree</i> yang berhasil dibangun
Alternate Course	-	-

5. Narasi *Use Case* Lihat Model *Tree*Tabel 6. 5 Narasi *Use Case* Lihat Model *Tree*

Lihat Model <i>Tree</i>		
Nama <i>Use Case</i>	Lihat Model <i>Tree</i>	
ID <i>Use Case</i>	5	
Aktor	User	
Deskripsi	<i>Use Case</i> ini merupakan proses bagaimana user dapat melihat model <i>tree</i> yang sudah berhasil dibentuk menggunakan metode <i>random forest</i>	
Kondisi Awal	Proses klasifikasi dengan metode <i>random forest</i> sudah berhasil dijalankan	
Kondisi Akhir	Model <i>tree</i> ditampilkan	
Typical Course	Aksi Aktor	Reaksi Sistem
	1) Memilih model klasifikasi pada “ <i>combo box</i> ” model	
		2) Menampilkan daftar model <i>tree</i> yang dimiliki oleh model yang dipilih user
	3) Memilih model <i>tree</i> pada “ <i>combo box</i> ” <i>tree</i>	
		4) Menampilkan model <i>tree</i> yang dipilih oleh user
Alternate Course	-	-

6. Narasi *Use Case* Uji Data Tunggal

Tabel 6. 6 Narasi Use Case Uji Data Tunggal

Uji Data Tunggal		
Nama Use Case	Uji Data Tunggal	
ID Use Case	6	
Aktor	User	
Deskripsi	<i>Use case</i> ini merupakan proses pengujian data tunggal menggunakan data atribut yang dimasukan oleh user	
Kondisi Awal	Proses klasifikasi menggunakan metode <i>random forest</i> sudah berhasil di jalankan dan user berada pada halaman “ <i>Single Data Test</i> ”	
Kondisi Akhir	Sistem menampilkan hasil klasifikasi dari data dengan nilai atribut yang sudah dimasukan oleh user	
Typical Course	Aksi Aktor	Reaksi Sistem
	1) Memasukan semua nilai atribut yang dibutuhkan pada <i>field</i> yang tersedia, lalu klik “Enter”	
		2) Menampilkan hasil klasifikasi pada halaman <i>Single Data Test</i>
Alternate Course	-	-

LAMPIRAN II: PROSEDUR PENGUJIAN DAN KASUS UJI

Tabel 6. 7 Prosedur Pengujian dan Kasus Uji

Use Case	Deskripsi	Presedur Pengujian	Masukan	Keluaran yang diharapkan	Hasil yang didapat	Catatan Proses Pengembangan
Input Dataset	Pengujian memasukan data dari <i>file</i> bertipe .csv	1) Jalankan sistem 2) Pilih menu <i>Preprocessing</i> tekan tombol cari	fileTest.csv	Data pada fileTest.csv ditampilkan kedalam tabel data	Data pada fileTest.csv ditampilkan kedalam tabel data	Tidak diperbaiki
	Pengujian memasukan data dari <i>file</i> selain bertipe .csv	3) Pilih file yang akan digunakan 4) Klik tombol “OK”	fileTest.xls	Muncul pesan error tipe file yang dimasukan tidak sesuai	Muncul pesan error tipe file yang dimasukan tidak sesuai	Tidak diperbaiki
Seleksi Atribut	Pengujian menghapus atribut dengan memilih satu persatu atribut	1) Tabel data pada halaman preprocessing sudah terisi data 2) Klik salah satu atribut yang ingin dihapus 3) Klik tombol “Buang”	Atribut yang dipilih: solo_killDeathRatio	Atribut solo_killDeathRatio berpindah dari list atribut yang digunakan ke list atribut yang dibuang	Atribut solo_killDeathRatio berpindah dari list atribut yang digunakan ke list atribut yang dibuang	Tidak diperbaiki
	Pengujian menghapus atribut dengan	1) Tabel data pada halaman	Atribut yang dipilih:	Atribut solo_killDeathRatio, solo_Losses,	Atribut solo_killDeathRatio,	Tidak diperbaiki

	memilih lebih dari satu atribut sekaligus	<p>preprocessing sudah terisi data</p> <p>2) Klik lebih dari satu atribut yang ingin dihapus dengan menahan tombol “Ctrl”</p> <p>3) Klik tombol “Buang”</p>	solo_killDeathRatio, solo_Losses, solo_WinTop10Ratio	solo_WinTop10Ratio berpindah dari list atribut yang digunakan ke list atribut yang dibuang	solo_Losses, solo_WinTop10Ratio berpindah dari list atribut yang digunakan ke list atribut yang dibuang	
	Pengujian membatalkan pemilihan atribut	<p>1) List atribut yang dibuang sudah berisi</p> <p>2) Klik atribut yang ingin batalkan dari penghapusan</p> <p>3) Klik tombol “Simpan”</p>	Atribut yang dipilih untuk dibatalkan: solo_killDeathRatio	Atribut solo_killDeathRatio berpindah dari list atribut yang dibuang ke list atribut yang digunakan	Atribut solo_killDeathRatio berpindah dari list atribut yang dibuang ke list atribut yang digunakan	Tidak diperbaiki
Inisialisasi Model	Pengujian memasukan jumlah <i>tree</i> dan jumlah <i>k-fold</i> dengan angka	<p>1) Proses preprocessing sudah dilakukan</p> <p>2) Masuk menu “Initialization”</p> <p>3) Inputkan angka pada field “jumlah k-fold” dan “jumlah <i>tree</i>”</p>	Masukan angka 50 pada field “jumlah <i>Tree</i> ” dan 3 pada field “Jumlah K-Fold”	field “jumlah <i>Tree</i> ” terisi angka 50 dan field “Jumlah K-Fold” terisi angka 3	field “jumlah <i>Tree</i> ” terisi angka 50 dan field “Jumlah K-Fold” terisi angka 3	Tidak diperbaiki
	Pengujian memasukan	1) Proses preprocessing sudah dilakukan	Masukan angka “a” pada field	field “jumlah <i>Tree</i> ” tidak terisi	field “jumlah <i>Tree</i> ” tidak	Tidak diperbaiki

	jumlah <i>tree</i> dan jumlah <i>k-fold</i> dengan bukan angka	2) Masuk menu “Initializatiom” 3) Inputksn huruf pada <i>field</i> “jumlah k-fold” dan “jumlah <i>tree</i> ”	“jumlah <i>Tree</i> ” dan “b” pada <i>field</i> “Jumlah K-Fold”	apapun dan <i>field</i> “Jumlah K-Fold” tidak terisi apapun juga	terisi apapun dan <i>field</i> “Jumlah K-Fold” tidak terisi apapun juga	
Proses Klasifikasi Dengan Metode <i>Random Forest</i>	Pengujian proses klasifikasi dengan mengisi jumlah <i>k-fold</i> dan <i>tree</i>	1) Proses preprocessing sudah dilakukan 2) Masuk menu “Initializatiom” 3) Inputksn nilai pada <i>field</i> “jumlah k-fold” dan “jumlah <i>tree</i> ” 4) Klik tombol “Run Algorithm”	Masukan angka 50 pada <i>field</i> “jumlah <i>Tree</i> ” dan 3 pada <i>field</i> “Jumlah K-Fold”	Proses klasifikasi berjalan	Proses klasifikasi berjalan	Tidak diperbaiki
	Pengujian proses klasifikasi dengan tanpa mengisi jumlah <i>k-fold</i> dan <i>tree</i>	1) Proses preprocessing sudah dilakukan 2) Masuk menu “Initializatiom” 3) Kosongkan <i>field</i> “jumlah k-fold” dan “jumlah <i>tree</i> ” 4) Klik tombol “Run Algorithm”	Kosongkan <i>field</i> “jumlah k-fold” dan “jumlah <i>tree</i> ”	Muncul pesan error masukan tidak sesuai	Muncul pesan error masukan tidak sesuai	Tidak diperbaiki

Lihat Model Tree	Pengujian penampilan model tree	<ol style="list-style-type: none"> 1) Proses klasifikasi sudah dilakukan 2) Model <i>tree</i> sudah ditampilkan 3) Klik “Combo Box” Model, pilih model 4) Klik “Combo Box” Tree, pilih <i>tree</i> 	Pilih Model 1 dan <i>tree</i> 2	Tampil model <i>tree</i> pada model 1	Tampil model <i>tree</i> pada model 1	Tidak diperbaiki
Uji Data Tunggal	Pengujian uji data tunggal dengan nilai atribut berupa angka	<ol style="list-style-type: none"> 1) Proses klasifikasi sudah dilakukan 2) Model <i>tree</i> sudah ditampilkan 3) Klik menu “Single Data Test” 4) Masukan nilai berupa angka pada field atribut yang tersedia 5) Klik “Enter” 	Masukan nilai berupa angka pada field atribut yang tersedia	Semua field atribut dapat terisi sesuai dengan nilai yang di masukan	Semua field atribut dapat terisi sesuai dengan nilai yang di masukan	Tidak diperbaiki
	Pengujian uji data tunggal dengan nilai atribut berupa bukan angka	<ol style="list-style-type: none"> 1) Proses klasifikasi sudah dilakukan 2) Model <i>tree</i> sudah ditampilkan 	Masukan nilai berupa huruf pada field atribut yang tersedia	Semua field atribut tidak terisi apapun	Semua field atribut tidak terisi apapun	Tidak diperbaiki

		3) Klik menu “Single Data Test”				
		4) Masukkan nilai berupa huruf pada field atribut yang tersedia				
		5) Klik “Enter”				



LAMPIRAN III: HITUNG MANUAL PENAMBANGAN DATA

Dalam perhitungan manual ini menggunakan dataset yang sama seperti yang digunakan perhitungan sistem. Dataset tersebut dapat dilihat pada tabel 6.8 serta pembagian foldnya. Untuk pembagian data testing dan data training pada setiap model dapat dilihat pada tabel 6.9.

Tabel 6. 8 Dataset yang digunakan untuk membangun model

	solo_WinRatio	solo_KillDeathRatio	solo_DamagePg	solo_HeadshotKillsPg	solo_MoveDistancePg	solo_TimeSurvivedPg
Fold 1	1	1,23	151,25	0,18	3653,76	1048,71
	0	1,83	193	0,38	1358,2	920,77
	2	2,01	225,11	0,44	3039,84	1004,96
	2	2,1	217,86	0,5	4970,75	1353,75
	0	1,02	124,03	0,14	2731,76	999,58
	1	1,08	123,32	0,22	3701,09	1077,55
Fold 2	0	1,04	129,73	0,23	2737,54	970,17
	1	1,05	129,18	0,25	2863,96	867,6
	2	2,03	210,54	0,59	3986,74	1171,84
	0	1,53	189,73	0,34	2170,23	819,47
	1	2,14	241,63	0,58	2178,41	804,51
	2	1,06	125,64	0,21	3312,12	1062,95

Tabel 6. 9 Pembagian Data Training dan Data Testing Setiap Model

Model 1	Training	Fold 1	Akurasi:	Akurasi Final:
	Testing	Fold 2		
Model 2	Training	Fold 2	Akurasi:	
	Testing	Fold 1		

A. Pembentukan *tree* model 1

Model 1 menggunakan fold 1 sebagai data training. Berikut adalah proses pembentukan *tree* pada model 1:

a. *Tree* 1

Random dataset untuk membangun *tree* 1 dapat dilihat pada tabel 6.10.

Tabel 6. 10 Random dataset untuk membangun *tree* 1 pada model 1

i	solo_WinRatio	solo_KillDeathRatio	solo_DamagePg	solo_HeadshotKillsPg	solo_MoveDistancePg	solo_TimeSurvivedPg
0	0	1,83	193	0,38	1358,2	920,77
1	0	1,02	124,03	0,14	2731,76	999,58
2	2	2,01	225,11	0,44	3039,84	1004,96
3	1	1,23	151,25	0,18	3653,76	1048,71
4	1	1,08	123,32	0,22	3701,09	1077,55
5	1	1,08	123,32	0,22	3701,09	1077,55

Berikut adalah langkah-langkah membangun *tree* 1:

1. Menentukan node 1

Hasil perhitungan information gain setiap atribut:

Atribut	Partisi	Total kasus	0	1	2	Entropy	Gain
Total		6	2	3	1	1,459148	
solo_KillDeathRatio							0,666667
	<1,23	3	1	2	0	0	
	>=1,23	3	1	1	1	1,584963	
solo_DamagePg							0,666667
	<151,25	3	1	2	0	0	
	>=151,25	3	1	1	1	1,584963	
solo_HeadshotKillsPg							0,666667
	<0,22	3	1	2	0	0	
	>=0,22	3	1	1	1	1,584963	
solo_MoveDistancePg							1,459148
	<3653,76	3	2	0	1	0	
	>=3653,76	3	0	3	0	0	
solo_TimeSurvivedPg							1,459148
	<1048,71	3	2	0	1	0	
	>=1048,71	3	0	3	0	0	

Atribut yang dipilih untuk dijadikan node: **solo_KillDeathRatio**

Hasil *tree* sementara:

```
└─ solo_MoveDistancePg: Root
   ├── (<3653,76)
   └── (>=3653,76)
```

*) Untuk mencari nilai split pada kolom partisi. Cukup mengambil indeks data (kolom i) ke-x. Dimana x adalah hasil pembagian jumlah kasus dibagi 2. Dengan kondisi data sudah diurutkan terlebih dahulu berdasar atribut yang akan dicari nilai splitnya

2. Menentukan node 1.1

i	solo_WinRatio	solo_KillDeathRatio	solo_DamagePg	solo_HeadshotKillsPg	solo_MoveDistancePg	solo_TimeSurvivedPg
0	0	1,83	193	0,38	1358,2	920,77
1	0	1,02	124,03	0,14	2731,76	999,58
2	2	2,01	225,11	0,44	3039,84	1004,96

Hasil perhitungan information gain setiap atribut:

Atribut	Partisi	Total kasus	0	1	2	Entropy	Gain
Total		3	2	0	1	0,918296	
solo_KillDeathRatio							0,251629
	<1,83	1	1	0	0		
	>=1,83	2	1	1	1		
solo_DamagePg							0,251629
	<193	1	1	0	0		
	>=193	2	1	1	1		
solo_HeadshotKillsPg							0,251629
	<0,38	1	1	0	0		
	>=0,38	2	1	1	1		
solo_MoveDistancePg							0,251629
	<2731,76	1	1	0	0		
	>=2731,76	2	1	1	1		

solo_TimeSurvivedPg							0,251629
	<999,58	1	1	-	0	0	
	>=999,58	2	1	-	1	1	

Attribut yang dipilih untuk dijadikan node: **solo_KillDeathRatio**

Hasil *tree* sementara:

```

└─ solo_MoveDistancePg: Root
  └─ (<3653,76) Node: solo_KillDeathRatio
    └─ (<1.83)
      └─ (>=1.83)
        └─ (>=3653,76)

```

3. Menentukan node 1.1.1

i	solo_WinRatio	solo_KillDeathRatio	solo_DamagePg	solo_HeadshotKillsPg	solo_MoveDistancePg	solo_TimeSurvivedPg
0	0	1,83	193	0,38	1358,2	920,77

Attribut yang dipilih untuk dijadikan node: **label kelas 0**

Hasil *tree* sementara:

```

└─ solo_MoveDistancePg: Root
   └─ (<3653,76) Node: solo_KillDeathRatio
      └─ (<1.83) Node: solo_WinRatio leaf: 0.0
         └─ (>=1.83)
            └─ (>=3653,76)

```

*) Setiap kali data tersisa satu maka atribut kelas akan dijadikan leaf node dengan nilai kelas yang tersisa

4. Menentukan node 1.1.2

i	solo_WinRatio	solo_KillDeathRatio	solo_DamagePg	solo_HeadshotKillsPg	solo_MoveDistancePg	solo_TimeSurvivedPg
0	0	1,83	193	0,38	1358,2	920,77
1	2	2,01	225,11	0,44	3039,84	1004,96

Hasil perhitungan information gain setiap atribut:

Atribut	Partisi	Total kasus	0	1	2	Entropy	Gain
Total		2	1	0	1	1	
solo_KillDeathRatio							1
	<2,01	1	1	-	0	0	
	>=2,01	1	0	-	1	0	
solo_DamagePg							1
	<225,11	1	1	-	0	0	
	>=225,11	1	0	-	1	0	
solo_HeadshotKillsPg							1
	<0,44	1	1	-	0	0	
	>=0,44	1	0	-	1	0	
solo_MoveDistancePg							1
	<3039,84	1	1	-	0	0	
	>=3039,84	1	0	-	1	0	

solo_TimeSurvivedPg						1
	<1004,96	1	1	-	0	
	>=1004,96	1	0	-	1	

Attribut yang dipilih untuk dijadikan node: **solo_KillDeathRatio**

Hasil *tree* sementara:

```

└─ solo_MoveDistancePg: Root
  └─ (<3653,76) Node: solo_KillDeathRatio
    └─ (<1.83) Node: solo_WinRatio leaf: 0
      └─ (>=1.83) Node: solo_KillDeathRatio
        └─ (<2.01)
          └─ (>=2.01)
            └─ (>=3653,76)

```

5. Menentukan node 1.1.2.1

i	solo_WinRatio	solo_KillDeathRatio	solo_DamagePg	solo_HeadshotKillsPg	solo_MoveDistancePg	solo_TimeSurvivedPg
0	0	1,83	193	0,38	1358,2	920,77

Attribut yang dipilih untuk dijadikan node: **Label kelas 0**

Hasil *tree* sementara:

```

└─ solo_MoveDistancePg: Root
  └─ (<3653,76) Node: solo_KillDeathRatio
    └─ (<1.83) Node: solo_WinRatio leaf: 0
    └─ (>=1.83) Node: solo_KillDeathRatio
      └─ (<2.01) Node: solo_WinRatio leaf: 0
      └─ (>=2.01)
        └─ (>=3653,76)

```

6. Menentukan node 1.1.2.2

i	solo_WinRatio	solo_KillDeathRatio	solo_DamagePg	solo_HeadshotKillsPg	solo_MoveDistancePg	solo_TimeSurvivedPg
0	2	2,01	225,11	0,44	3039,84	1004,96

Attribut yang dipilih untuk dijadikan node: **Label kelas 2**

Hasil *tree* sementara:

```

└─ solo_MoveDistancePg: Root
  └─ (<3653,76) Node: solo_KillDeathRatio
    └─ (<1.83) Node: solo_WinRatio leaf: 0
    └─ (>=1.83) Node: solo_KillDeathRatio
      └─ (<2.01) Node: solo_WinRatio leaf: 0
      └─ (>=2.01) Node: solo_WinRatio leaf: 2
    └─ (>=3653,76) Node: solo_WinRatio leaf: 1
  
```


7. Menentukan node 1.2

i	solo_WinRatio	solo_KillDeathRatio	solo_DamagePg	solo_HeadshotKillsPg	solo_MoveDistancePg	solo_TimeSurvivedPg
0	1	1,23	151,25	0,18	3653,76	1048,71
1	1	1,08	123,32	0,22	3701,09	1077,55
2	1	1,08	123,32	0,22	3701,09	1077,55

Attribut yang dipilih untuk dijadikan node: **Label kelas 1**

Hasil akhir *tree*:

```

└─ solo_MoveDistancePg: Root
  └─ (<3653,76) Node: solo_KillDeathRatio
    └─ (<1.83) Node: solo_WinRatio leaf: 0
    └─ (>=1.83) Node: solo_KillDeathRatio
      └─ (<2.01) Node: solo_WinRatio leaf: 0
      └─ (>=2.01) Node: solo_WinRatio leaf: 2
    └─ (>=3653,76) Node: solo_WinRatio leaf: 1
  
```

*) Setiap kali data yang tersisa memiliki nilai atribut kelas yang sama, maka atribut kelas dijadikan leaf node dengan nilai kelas tersebut.

b. *Tree 2*

Random dataset untuk membangun *tree 2* dapat dilihat pada tabel 6.11

Tabel 6. 11 Random dataset untuk membangun *tree 2* pada model 1

i	solo_WinRatio	solo_KillDeathRatio	solo_DamagePg	solo_HeadshotKillsPg	solo_MoveDistancePg	solo_TimeSurvivedPg
0	0	1,02	124,03	0,14	2731,76	999,58
1	1	1,23	151,25	0,18	3653,76	1048,71
2	1	1,23	151,25	0,18	3653,76	1048,71
3	0	1,83	193	0,38	1358,2	920,77
4	0	1,83	193	0,38	1358,2	920,77
5	2	2,1	217,86	0,5	4970,75	1353,75

Berikut adalah langkah-langkah membangun *tree* 2:

1. Menentukan node 1

Hasil perhitungan information gain setiap atribut:

Atribut	Partisi	Total kasus	0	1	2	Entropy	Gain
Total		6	3	2	1	1,459148	
solo_KillDeathRatio							1,459148
	<1,83	3	1	2	0	0	
	>=1,83	3	2	0	1	0	
solo_DamagePg							1,459148
	<193	3	1	2	0	0	
	>=193	3	2	0	1	0	
solo_HeadshotKillsPg							1,459148
	<0,38	3	1	2	0	0	
	>=0,38	3	0	1	2	0	
solo_MoveDistancePg							1,459148
	<3653,76	3	3	0	0	0	
	>=3653,76	3	0	2	1	0	
solo_TimeSurvivedPg							1,459148
	<1048,71	3	3	0	0	0	
	>=1048,71	3	0	2	1	0	

Attribut yang dipilih untuk dijadikan node: **solo_KillDeathRatio**

Hasil *tree* sementara:

└─ solo_KillDeathRatio: Root
 └─ (<1,83)
 └─ (>=1,83)

2. Menentukan node 1.1

i	solo_WinRatio	solo_KillDeathRatio	solo_DamagePg	solo_HeadshotKillsPg	solo_MoveDistancePg	solo_TimeSurvivedPg
0	0	1,02	124,03	0,14	2731,76	999,58
1	1	1,23	151,25	0,18	3653,76	1048,71
2	1	1,23	151,25	0,18	3653,76	1048,71

Hasil perhitungan information gain setiap atribut:

Atribut	Partisi	Total kasus	0	1	2	Entropy	Gain
Total		3	1	2	0	0,918296	
solo_KillDeathRatio							0,918296
	<1,23	1	1	0	-	0	
	>=1,23	2	0	2	-	0	
solo_DamagePg							0,918296
	<151,25	1	1	0	-	0	
	>=151,25	2	0	2	-	0	
solo_HeadshotKillsPg							0,918296
	<0,18	1	1	0	-	0	
	>=0,18	2	0	2	-	0	
solo_MoveDistancePg							0,918296
	<3653,76	1	1	0	-	0	
	>=3653,76	2	0	2	-	0	
solo_TimeSurvivedPg							0,918296
	<1048,71	1	1	0	-	0	
	>=1048,71	2	0	2	-	0	

Atribut yang dipilih untuk dijadikan node: **solo_KillDeathRatio**

Hasil *tree* sementara:

```

└─ solo_KillDeathRatio: Root
  └─ (<1.83) Node: solo_KillDeathRatio
    │ │ └─ (<1,23)
    │ │ └─ (>=1,23)
    └─ (>=1,83)
  
```

3. Menentukan node 1.1.1

i	solo_WinRatio	solo_KillDeathRatio	solo_DamagePg	solo_HeadshotKillsPg	solo_MoveDistancePg	solo_TimeSurvivedPg
0	0	1,02	124,03	0,14	2731,76	999,58

Attribut yang dipilih untuk dijadikan node: **Label kelas 0**

Hasil *tree* sementara:

```

└─ solo_KillDeathRatio: Root
  └─ (<1.83) Node: solo_KillDeathRatio
    │ │ └─ (<1,23) Node: solo_WinRatio leaf: 0
    │ │ └─ (>=1,23)
    └─ (>=1,83)
  
```

4. Menentukan node 1.1.2

i	solo_WinRatio	solo_KillDeathRatio	solo_DamagePg	solo_HeadshotKillsPg	solo_MoveDistancePg	solo_TimeSurvivedPg
1	1	1,23	151,25	0,18	3653,76	1048,71
2	1	1,23	151,25	0,18	3653,76	1048,71

Attribut yang dipilih untuk dijadikan node: **Label kelas 1**

Hasil *tree* sementara:

```

└─ solo_KillDeathRatio: Root
   └─ (<1.83) Node: solo_KillDeathRatio
      └─ (<1,23) Node: solo_WinRatio leaf: 0
         └─ (≥1,23) Node: solo_WinRatio leaf: 1
      └─ (≥1,83)

```

5. Menentukan node 1.2

i	solo_WinRatio	solo_KillDeathRatio	solo_DamagePg	solo_HeadshotKillsPg	solo_MoveDistancePg	solo_TimeSurvivedPg
0	0	1,83	193	0,38	1358,2	920,77
1	0	1,83	193	0,38	1358,2	920,77
2	2	2,1	217,86	0,5	4970,75	1353,75

Hasil perhitungan information gain setiap atribut:

Atribut	Partisi	Total kasus	0	1	2	Entropy	Gain
Total		3	2	0	1	0,918296	
solo_KillDeathRatio							0
	<1,83	0	0	-	0	0	
	>=1,83	3	2	-	1	0,918296	
solo_DamagePg							0
	<193	0	0	-	0	0	
	>=193	3	2	-	1	0,918296	
solo_HeadshotKillsPg							0
	<0,38	0	0	-	0	0	
	>=0,38	3	2	-	1	0,918296	
solo_MoveDistancePg							0
	<1358,2	0	0	-	0	0	
	>=1358,2	3	2	-	1	0,918296	
solo_TimeSurvivedPg							0
	<920,77	0	0	-	0	0	
	>=920,77	3	2	-	1	0,918296	

Atribut yang dipilih untuk dijadikan node: **Label kelas 0**

Hasil *tree* sementara:

```

└─ solo_KillDeathRatio: Root
  └─ (<1,83) Node: solo_KillDeathRatio
    └─ (<1,23) Node: solo_WinRatio leaf: 0
    └─ (>=1,23) Node: solo_WinRatio leaf: 1
    └─ (>=1,83) Node: solo_WinRatio leaf: 0
  
```

*) Jika ada atribut yang terpilih menghasilkan 1 partisi saja, maka nilai kelas label yang mendominasi dijadikan leaf node. Dalam contoh kasus ini adalah atribut **solo_KillDeathRatio** dipilih untuk dijadikan node, namun atribut tersebut menghasilkan salah satu partisi bernilai nol yaitu partisi $<1,83$ maka nilai kelas label 0 (yang mendominasi) dipilih untuk dijadikan leaf node

B. Pembentukan *tree* model 2

Model 2 menggunakan fold 2 sebagai data training. Berikut adalah proses pembentukan *tree* pada model 2:

a. *Tree* 1

Random dataset untuk membangun *tree* 1 dapat dilihat pada tabel 6.12.

Tabel 6. 12 Random dataset untuk membangun *tree* 1 pada model 2

i	solo_WinRatio	solo_KillDeathRatio	solo_DamagePg	solo_HeadshotKillsPg	solo_MoveDistancePg	solo_TimeSurvivedPg
0	2	1,06	125,64	0,21	3312,12	1062,95
1	2	1,06	125,64	0,21	3312,12	1062,95
2	1	1,05	129,18	0,25	2863,96	867,6
3	1	1,05	129,18	0,25	2863,96	867,6
4	0	1,53	189,73	0,34	2170,23	819,47
5	1	2,14	241,63	0,58	2178,41	804,51

Berikut adalah langkah-langkah membangun *tree* 1:

1. Menentukan node 1

Hasil perhitungan information gain setiap atribut:

Atribut	Partisi	Total kasus	0	1	2	Entropy	Gain
Total		6	1	3	2	1,459148	
solo_KillDeathRatio							0,459148
	<1,06	2	0	2	0	0	
	>=1,06	4	1	1	2	1,5	
solo_DamagePg							1,459148
	<129,18	2	0	0	2	0	
	>=129,18	4	1	3	0	0	
solo_HeadshotKillsPg							1,459148
	<0,25	2	0	0	2	0	
	>=0,25	4	1	3	0	0	
solo_MoveDistancePg							1,459148
	<2863,96	2	1	1	0	0	
	>=2863,96	4	0	2	2	0	
solo_TimeSurvivedPg							1,459148
	<867,6	2	1	1	0	0	
	>=867,6	4	0	2	2	0	

Attribut yang dipilih untuk dijadikan node: **solo_DamagePg**

Hasil *tree* sementara:

```
└─ solo_DamagePg: Root
  └─ (<129,18)
    └─ (>=129,18)
```

2. Menentukan node 1.1

i	solo_WinRatio	solo_KillDeathRatio	solo_DamagePg	solo_HeadshotKillsPg	solo_MoveDistancePg	solo_TimeSurvivedPg
0	2	1,06	125,64	0,21	3312,12	1062,95
1	2	1,06	125,64	0,21	3312,12	1062,95

Attribut yang dipilih untuk dijadikan node: **Label kelas 2**

Hasil *tree* sementara:

```

└─ solo_DamagePg: Root
  └─ (<129,18) Node: solo_WinRatio leaf: 2
    └─ (>=129,18)
  
```

3. Menentukan node 1.2

i	solo_WinRatio	solo_KillDeathRatio	solo_DamagePg	solo_HeadshotKillsPg	solo_MoveDistancePg	solo_TimeSurvivedPg
0	1	1,05	129,18	0,25	2863,96	867,6
1	1	1,05	129,18	0,25	2863,96	867,6
2	0	1,53	189,73	0,34	2170,23	819,47
3	1	2,14	241,63	0,58	2178,41	804,51

Hasil perhitungan information gain setiap atribut:

Atribut	Partisi	Total kasus	0	1	2	Entropy	Gain
Total		4	1	3	0	0,811278	
solo_KillDeathRatio							0,561278
	<1,53	2	0	2	-	0	
	>=1,53	2	1	1	-	0,5	
solo_DamagePg							0,561278
	<189,73	2	0	2	-	0	
	>=189,73	2	1	1	-	0,5	
solo_HeadshotKillsPg							0,561278
	<0,34	2	0	2	-	0	
	>=0,34	2	1	1	-	0,5	
solo_MoveDistancePg							0,561278
	<2863,96	2	1	1	-	0,5	
	>=2863,96	2	0	2	-	0	

solo_TimeSurvivedPg							0,561278
	<867,6	2	1	1	-	0,5	
	>=867,6	2	0	2	-	0	

Attribut yang dipilih untuk dijadikan node: **solo_KillDeathRatio**

Hasil *tree* sementara:



4. Menentukan node 1.2.1

i	solo_WinRatio	solo_KillDeathRatio	solo_DamagePg	solo_HeadshotKillsPg	solo_MoveDistancePg	solo_TimeSurvivedPg
0	1	1,05	129,18	0,25	2863,96	867,6
1	1	1,05	129,18	0,25	2863,96	867,6

Attribut yang dipilih untuk dijadikan node: **Label kelas 1**

Hasil *tree* sementara:

```

└─ solo_DamagePg: Root
  └─ (<129,18) Node: solo_WinRatio leaf: 2
    └─ (>=129,18) Node: solo_KillDeathRatio
      └─ (<1.53) Node: solo_WinRatio leaf: 2
        └─ (>=1.53)

```

5. Menentukan node 1.2.2

i	solo_WinRatio	solo_KillDeathRatio	solo_DamagePg	solo_HeadshotKillsPg	solo_MoveDistancePg	solo_TimeSurvivedPg
0	0	1,53	189,73	0,34	2170,23	819,47
1	1	2,14	241,63	0,58	2178,41	804,51

Hasil perhitungan information gain setiap atribut:

Atribut	Partisi	Total kasus	0	1	2	Entropy	Gain
Total		2	1	1	0	1	
solo_KillDeathRatio							1
	<2,14	1	1	0	-	0	
	>=2,14	1	0	1	-	0	
solo_DamagePg							1
	<241,63	1	1	0	-	0	
	>=241,63	1	0	1	-	0	
solo_HeadshotKillsPg							1
	<0,58	1	1	0	-	0	
	>=0,58	1	0	1	-	0	
solo_MoveDistancePg							1
	<2170,23	1	1	0	-	0	
	>=2170,23	1	0	1	-	0	
solo_TimeSurvivedPg							1
	<804,51	1	0	1	-	0	
	>=804,51	1	1	0	-	0	

Attribut yang dipilih untuk dijadikan node: **solo_KillDeathRatio**

Hasil *tree* sementara:

```

└─ solo_DamagePg: Root
  └─ (<129,18) Node: solo_WinRatio leaf: 2
    └─ (>=129,18) Node: solo_KillDeathRatio
      └─ (<1.53) Node: solo_WinRatio leaf: 2
        └─ (>=1.53) Node: solo_KillDeathRatio
          └─ (<2,14)
            └─ (>=2,14)
  
```

6. Menentukan node 1.2.2.1

i	solo_WinRatio	solo_KillDeathRatio	solo_DamagePg	solo_HeadshotKillsPg	solo_MoveDistancePg	solo_TimeSurvivedPg
0	0	1,53	189,73	0,34	2170,23	819,47

Attribut yang dipilih untuk dijadikan node: **Label kelas 0**

Hasil *tree* sementara:

```

└─ solo_DamagePg: Root
  └─ (<129,18) Node: solo_WinRatio leaf: 2
    └─ (>=129,18) Node: solo_KillDeathRatio
      └─ (<1.53) Node: solo_WinRatio leaf: 2
        └─ (>=1.53) Node: solo_KillDeathRatio
          └─ (<2,14) Node: solo_WinRatio leaf: 0
            └─ (>=2,14)

```

7. Menentukan node 1.2.2.2

i	solo_WinRatio	solo_KillDeathRatio	solo_DamagePg	solo_HeadshotKillsPg	solo_MoveDistancePg	solo_TimeSurvivedPg
0	1	2,14	241,63	0,58	2178,41	804,51

Attribut yang dipilih untuk dijadikan node: **Label kelas 1**

Hasil akhir *tree* 2:

```

└─ solo_DamagePg: Root
  └─ (<129,18) Node: solo_WinRatio leaf: 2
    └─ (>=129,18) Node: solo_KillDeathRatio
      └─ (<1.53) Node: solo_WinRatio leaf: 2
        └─ (>=1.53) Node: solo_KillDeathRatio
          └─ (<2,14) Node: solo_WinRatio leaf: 0
            └─ (>=2,14) Node: solo_WinRatio leaf: 1
  
```

b. *Tree 2*

Random dataset untuk membangun *tree 2* dapat dilihat pada tabel 6.13.

Tabel 6. 13 Random dataset untuk membangun *tree 2* pada model 2

i	solo_WinRatio	solo_KillDeathRatio	solo_DamagePg	solo_HeadshotKillsPg	solo_MoveDistancePg	solo_TimeSurvivedPg
0	1	2,14	241,63	0,58	2178,41	804,51
1	0	1,04	129,73	0,23	2737,54	970,17
2	0	1,04	129,73	0,23	2737,54	970,17
3	2	1,06	125,64	0,21	3312,12	1062,95
4	2	2,03	210,54	0,59	3986,74	1171,84
5	2	2,03	210,54	0,59	3986,74	1171,84

Berikut adalah langkah-langkah membangun *tree* 2:

1. Menentukan node 1

Hasil perhitungan information gain setiap atribut:

Atribut	Partisi	Total kasus	0	1	2	Entropy	Gain
Total		6	2	1	3	1,459148	
solo_KillDeathRatio							1,459148
	<2,03	2	2	0	0	0	
	>=2,03	4	0	1	3	0	
solo_DamagePg							0,459148
	<210,54	2	1	0	1	0	
	>=210,54	4	1	1	2	1,5	
solo_HeadshotKillsPg							0,459148
	<0,58	2	1	0	1	0	
	>=0,58	4	1	1	2	1,5	
solo_MoveDistancePg							1,459148
	<3312,12	2	1	1	0	0	
	>=3312,12	4	1	0	3	0	
solo_TimeSurvivedPg							1,459148
	<1062,95	2	1	1	0	0	
	>=1062,95	4	1	0	3	0	

Atribut yang dipilih untuk dijadikan node: **solo_KillDeathRatio**

Hasil *tree* sementara:

└─ solo_KillDeathRatio: Root
 └─ (<2,03)
 └─ (>=2,03)

2. Menentukan node 1.1

i	solo_WinRatio	solo_KillDeathRatio	solo_DamagePg	solo_HeadshotKillsPg	solo_MoveDistancePg	solo_TimeSurvivedPg
0	0	1,04	129,73	0,23	2737,54	970,17
1	0	1,04	129,73	0,23	2737,54	970,17
2	2	1,06	125,64	0,21	3312,12	1062,95

Hasil perhitungan information gain setiap atribut:

Atribut	Partisi	Total kasus	0	1	2	Entropy	Gain
Total		3	2	0	1	0,918296	
solo_KillDeathRatio							0,251629
	<1,04	1	1 -		0	0	
	>=1,04	2	1 -		1	1	
solo_DamagePg							0,918296
	<129,73	1	0 -		1	0	
	>=129,73	2	2 -		0	0	
solo_HeadshotKillsPg							0,918296
	<0,25	0	0 -		0	0	
	>=0,25	1	1 -		0	0	
solo_MoveDistancePg							0,584963
	<2737,54	1	1 -		0	0	
	>=2737,54	2	1 -		1	0,5	
solo_TimeSurvivedPg							0,918296
	<970,17	2	1 -		1	0	
	>=970,17	2	1 -		1	0	

Atribut yang dipilih untuk dijadikan node: **solo_DamagePg**

Hasil *tree* sementara:

```

└─ solo_KillDeathRatio: Root
  └─ (<2,03) Node: solo_DamagePg
    └─ (<129,18)
      └─ (>=129,18)
        └─ (>=2,03)
  └─ (>=2,03)
  
```

3. Menentukan node 1.1.1

i	solo_WinRatio	solo_KillDeathRatio	solo_DamagePg	solo_HeadshotKillsPg	solo_MoveDistancePg	solo_TimeSurvivedPg
0	0	1,04	129,73	0,23	2737,54	970,17
1	0	1,04	129,73	0,23	2737,54	970,17

Attribut yang dipilih untuk dijadikan node: **Label kelas 0**

Hasil *tree* sementara:

```

└─ solo_KillDeathRatio: Root
  └─ (<2,03) Node: solo_DamagePg
    └─ (<129,18) Node: solo_WinRatio leaf: 0
      └─ (>=129,18)
        └─ (>=2,03)
  
```

4. Menentukan node 1.1.2

i	solo_WinRatio	solo_KillDeathRatio	solo_DamagePg	solo_HeadshotKillsPg	solo_MoveDistancePg	solo_TimeSurvivedPg
0	2	1,06	125,64	0,21	3312,12	1062,95

Hasil perhitungan information gain setiap atribut:

Atribut yang dipilih untuk dijadikan node: **Label kelas 2**

Hasil *tree* sementara:

```

└─ solo_KillDeathRatio: Root
  └─ (<2,03) Node: solo_DamagePg
    └─ (<129,18) Node: solo_WinRatio leaf: 0
      └─ (>=129,18) Node: solo_WinRatio leaf: 2
    └─ (>=2,03)
  
```

5. Menentukan node 1.2

i	solo_WinRatio	solo_KillDeathRatio	solo_DamagePg	solo_HeadshotKillsPg	solo_MoveDistancePg	solo_TimeSurvivedPg
2	1	2,14	241,63	0,58	2178,41	804,51
0	2	2,03	210,54	0,59	3986,74	1171,84
1	2	2,03	210,54	0,59	3986,74	1171,84

Hasil perhitungan information gain setiap atribut:

Atribut	Partisi	Total kasus	0	1	2	Entropy	Gain
Total		3	0	1	2	0,918296	
solo_KillDeathRatio							0,251629
	<2,03	1	-	0	1	0	
	>=2,03	2	-	1	1	1	
solo_DamagePg							0,251629
	<210,54	1	-	0	1	0	
	>=210,54	2	-	1	1	1	
solo_HeadshotKillsPg							0,918296
	<0,59	1	-	1	0	0	
	>=0,59	2	-	0	2	0	
solo_MoveDistancePg							0,918296
	<3986,74	1	-	1	0	0	
	>=3986,74	2	-	0	2	0	
solo_TimeSurvivedPg							0,918296
	<1171,84	1	-	1	0	0	
	>=1171,84	0	-	0	0	0	

Atribut yang dipilih untuk dijadikan node: **solo_HeadshotKillsPg**

Hasil *tree* sementara:

```

└─ solo_KillDeathRatio: Root
  ├── (<2,03) Node: solo_DamagePg
  │   ├── (<129,18) Node: solo_WinRatio leaf: 0
  │   └── (>=129,18) Node: solo_WinRatio leaf: 2
  └── (>=2,03) Node: solo_HeadshotKillsPg
      ├── (<0,59)
      └── (>=0,59)
  
```

6. Menentukan node 1.2.1

i	solo_WinRatio	solo_KillDeathRatio	solo_DamagePg	solo_HeadshotKillsPg	solo_MoveDistancePg	solo_TimeSurvivedPg
0	1	2,14	241,63	0,58	2178,41	804,51

Attribut yang dipilih untuk dijadikan node: **Label kelas 1**

Hasil *tree* sementara:

```

└─ solo_KillDeathRatio: Root
  └─ (<2,03) Node: solo_DamagePg
    └─ (<129,18) Node: solo_WinRatio leaf: 0
      └─ (>=129,18) Node: solo_WinRatio leaf: 2
    └─ (>=2,03) Node: solo_HeadshotKillsPg
      └─ (<0,59) Node: solo_WinRatio leaf: 1
        └─ (>=0,59)
  
```

7. Menentukan node 1.2.2

i	solo_WinRatio	solo_KillDeathRatio	solo_DamagePg	solo_HeadshotKillsPg	solo_MoveDistancePg	solo_TimeSurvivedPg
0	2	2,03	210,54	0,59	3986,74	1171,84
1	2	2,03	210,54	0,59	3986,74	1171,84

Attribut yang dipilih untuk dijadikan node: **Label kelas 2**

Hasil akhir *tree* 2:

```

└─ solo_KillDeathRatio: Root
  └─ (<2,03) Node: solo_DamagePg
    └─ (<129,18) Node: solo_WinRatio leaf: 0
      └─ (>=129,18) Node: solo_WinRatio leaf: 2
    └─ (>=2,03) Node: solo_HeadshotKillsPg
      └─ (<0,59) Node: solo_WinRatio leaf: 1
        └─ (>=0,59) Node: solo_WinRatio leaf: 2
  
```

C. Pengujian Data Testing

a. Model 1

Model 1 menggunakan fold 2 sebagai data testing. Hasil klasifikasi data testing model 1 dapat dilihat pada tabel 6. Untuk *tree* 1 dan 2 dapat dilihat pada gambar 6. dan 6..

```

└─ solo_MoveDistancePg: Root
  └─ (<3653,76) Node: solo_KillDeathRatio
    └─ (<1.83) Node: solo_WinRatio leaf: 0
    └─ (>=1.83) Node: solo_KillDeathRatio
      └─ (<2.01) Node: solo_WinRatio leaf: 0
      └─ (>=2.01) Node: solo_WinRatio leaf: 2
    └─ (>=3653,76) Node: solo_WinRatio leaf: 1
  
```

Gambar 6. 1 Hasil Akhir *Tree* 1 Pada Model 1 Penghitungan Manual

```

└─ solo_KillDeathRatio: Root
  └─ (<1.83) Node: solo_KillDeathRatio
    └─ (<1,23) Node: solo_WinRatio leaf: 0
    └─ (>=1,23) Node: solo_WinRatio leaf: 1
  └─ (>=1,83) Node: solo_WinRatio leaf: 0
  
```

Gambar 6. 2 Hasil Akhir *Tree* 2 Pada Model 1 Penghitungan Manual

Tabel 6. 14 Hasil Klasifikasi Data Testing Model 1

solo_Win Ratio	solo_KillDeat hRatio	solo_Damage Pg	solo_Headshot KillsPg	solo_Move DistancePg	solo_Time SurvivedPg	Voting		Hasil voting
						Tree 1	Tree 2	
0	1,04	129,73	0,23	2737,54	970,17	0	0	0
1	1,05	129,18	0,25	2863,96	867,6	0	0	0
2	2,03	210,54	0,59	3986,74	1171,84	1	0	1
0	1,53	189,73	0,34	2170,23	819,47	0	1	0
1	2,14	241,63	0,58	2178,41	804,51	2	0	2
2	1,06	125,64	0,21	3312,12	1062,95	0	0	0

Tabel 6. 15 Evaluasi Hasil Model 1

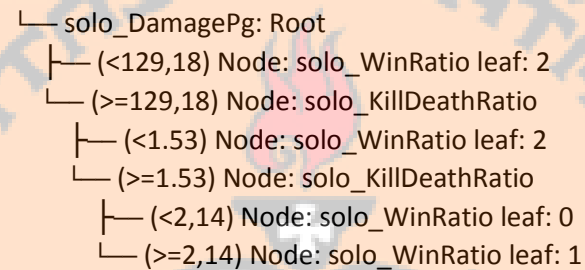
		Kelas Prediksi		
		Kelas=0	Kelas=1	Kelas=2
Kelas Aktual	Kelas=0	2	0	0
	Kelas=1	1	0	1
	Kelas=2	1	1	0

Untuk memperoleh akurasi model 1 maka dapat menggunakan rumus (2.5), berikut adalah hasil perhitungannya:

$$Akurasi = \frac{(2 + 0 + 0)}{(2 + 0 + 0) + (1 + 0 + 1) + (1 + 1 + 0)} \times 100 = 33,33$$

b. Model 2

Model 2 menggunakan fold 1 sebagai data testing. Hasil klasifikasi data testing model 2 dapat dilihat pada tabel 6.

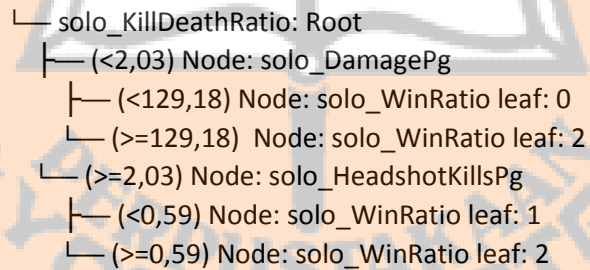


```

graph TD
    Root[solo_DamagePg: Root] -- "( <129,18 )" --> Node1[solo_WinRatio leaf: 2]
    Root -- "( >=129,18 )" --> Node2[solo_KillDeathRatio]
    Node2 -- "( <1.53 )" --> Node3[solo_WinRatio leaf: 2]
    Node2 -- "( >=1.53 )" --> Node4[solo_KillDeathRatio]
    Node4 -- "( <2,14 )" --> Node5[solo_WinRatio leaf: 0]
    Node4 -- "( >=2,14 )" --> Node6[solo_WinRatio leaf: 1]
  
```

└ solo_DamagePg: Root
 └ (<129,18) Node: solo_WinRatio leaf: 2
 └ (>=129,18) Node: solo_KillDeathRatio
 └ (<1.53) Node: solo_WinRatio leaf: 2
 └ (>=1.53) Node: solo_KillDeathRatio
 └ (<2,14) Node: solo_WinRatio leaf: 0
 └ (>=2,14) Node: solo_WinRatio leaf: 1

Gambar 6. 3 Hasil Akhir *Tree* 1 Pada Model 2 Penghitungan Manual



```

graph TD
    Root[solo_KillDeathRatio: Root] -- "( <2,03 )" --> Node1[solo_DamagePg]
    Node1 -- "( <129,18 )" --> Node2[solo_WinRatio leaf: 0]
    Node1 -- "( >=129,18 )" --> Node3[solo_WinRatio leaf: 2]
    Root -- "( >=2,03 )" --> Node4[solo_HeadshotKillsPg]
    Node4 -- "( <0,59 )" --> Node5[solo_WinRatio leaf: 1]
    Node4 -- "( >=0,59 )" --> Node6[solo_WinRatio leaf: 2]
  
```

└ solo_KillDeathRatio: Root
 └ (<2,03) Node: solo_DamagePg
 └ (<129,18) Node: solo_WinRatio leaf: 0
 └ (>=129,18) Node: solo_WinRatio leaf: 2
 └ (>=2,03) Node: solo_HeadshotKillsPg
 └ (<0,59) Node: solo_WinRatio leaf: 1
 └ (>=0,59) Node: solo_WinRatio leaf: 2

Gambar 6. 4 Hasil Akhir *Tree* 2 Pada Model 2 Penghitungan Manual

Tabel 6. 16 Hasil Klasifikasi Data Testing Model 2

solo_Win Ratio	solo_KillDeath Ratio	solo_Damage Pg	solo_Headshot KillsPg	solo_Move DistancePg	solo_Time SurvivedPg	Voting		Hasil voting
						Tree 1	Tree 2	
1	1,23	151,25	0,18	3653,76	1048,71	2	2	2
0	1,83	193	0,38	1358,2	920,77	0	2	0
2	2,01	225,11	0,44	3039,84	1004,96	0	2	0
2	2,1	217,86	0,5	4970,75	1353,75	0	1	0
0	1,02	124,03	0,14	2731,76	999,58	2	0	2
1	1,08	123,32	0,22	3701,09	1077,55	2	0	2

Tabel 6. 17 Evaluasi Hasil Model 1

	Kelas Prediksi			
		Kelas=0	Kelas=1	Kelas=2
	Kelas=0	1	0	1
	Kelas=1	0	0	2
	Kelas=2	2	0	0

Untuk memperoleh akurasi model 2 maka dapat menggunakan rumus (2.5), berikut adalah hasil perhitungannya:

$$Akurasi = \frac{(1+0+0)}{(1+0+1)+(0+0+2)+(2+0+0)} \times 100 = 16,67$$

D. Akurasi Final

Untuk mendapatkan akurasi final maka diambil rata-rata akurasi yang didapat pada setiap model, dalam kasus ini ada dua model. Berikut adalah perhitungan akurasi final:

$$Akurasi = \frac{33,33 + 16,67}{2} = 25$$

Dari hasil penghitungan di atas diperoleh akurasi final sebesar 25%.