LOGISTIC REGRESSION DENGAN *MULTI-SWARM OPTIMIZATION* UNTUK MEMPREDIKSI KEMENANGAN DALAM PERMAINAN DOTA 2

Raden Hasbie Finno Hadiyanto[1], Galih Hermawan[2]

[1,2] Program Studi Teknik Informatika, Universitas Komputer Indonesia
Jl. Dipati Ukur No.112-116, Bandung 40132
E-mail : rdhasbiefh@gmail.com[1], galih.hermawan@email.unikom.ac.id[2]

ABSTRAK

Defend of The Ancients 2 atau DOTA 2 adalah sebuah permainan dimana permainan terdiri dari 2 tim, yaitu *DIRE* dan *RADIANT* yang masing-masing beranggotakan 5 orang pemain. Dalam permainan ini, kemenangan ditentukan oleh tim mana yang lebih dulu dapat menghancurkan *ancient* lawan. Terdapat beberapa faktor yang dapat mempengaruhi hasil akhir dari permainan, salah satunya adalah gaya bermain tim. Faktor tersebut dapat dijadikan sebagai acuan untuk memprediksi tim mana yang akan memenangkan permainan. Untuk melakukan prediksi, dalam penelitian ini digunakan metode *Logistic Regression*. Sebelum prediksi dilakukan, terlebih dahulu akan dilakukan pencarian solusi optimal berupa bobot terbaik yang akan digunakan dalam perhitungan *Logistic Regression*. Untuk mendapatkan solusi optimal tersebut, dalam penelitian ini digunakan metode *Multi-Swarm Optimization*. Dengan mengkombinasikan kedua metode tersebut, solusi yang optimal diperoleh dengan menggunakan 3 *swarm*, 15 partikel, 40 iterasi, batas ruang pencarian (-10,10), dan batas kecepatan partikel (-1,1), dimana tingkat akurasi yang dihasilkan sangat baik, yaitu sebesar 95,76% untuk tingkat akurasi terendah dan 97,51% untuk tingkat akurasi tertinggi. Hasil tersebut menunjukkan bahwa metode yang digunakan dapat menghasilkan solusi yang optimal.

Kata kunci : *Defend of The Ancients 2*, *Logistic Regression*, *Multi-Swarm Optimization*, Prediksi Kemenangan, *Game*

1. PENDAHULUAN

1.1 Latar Belakang

Defend of The Ancients 2 atau yang lebih dikenal dengan DOTA 2 adalah sebuah permainan dengan *genre* MOBA (*Multiplayer Online Battle Arena*) dimana permainan terdiri dari 2 tim, yaitu *DIRE* dan *RADIANT* yang masing-masing beranggotakan 5 orang pemain dengan setiap pemainnya memainkan karakter (*hero*) yang berbeda-beda. Kedua tim akan saling menghancurkan markas satu sama lain dan mempertahankan markas nya masing-masing. Kemenangan akan ditentukan oleh tim mana yang lebih dulu dapat menghancurkan *ancient* pada markas lawan.

Dalam permainan ini, terdapat beberapa faktor yang dapat mempengaruhi hasil akhir dari suatu permainan seperti dari faktor pemilihan karakter dan gaya bermain tim [1]. Faktor-faktor tersebut dapat dijadikan sebagai acuan untuk memprediksi tim mana yang akan memenangkan permainan berdasarkan data-data pertandingan yang telah dilakukan sebelumnya.

Terdapat beberapa penelitian yang telah dilakukan untuk memprediksi kemenangan dalam permainan DOTA 2. Zhengyao Li, Dingyue Cui, dan Chen Li telah melakukan penelitian untuk memprediksi kemenangan dalam permainan DOTA 2 dengan membandingkan beberapa algoritma berdasarkan gaya bermain tim dan menghasilkan *Logistic Regression* (*LR*) sebagai model terbaik dengan tingkat akurasi sebesar 59,713% [2]. Pada penelitian lainnya, Nicholas Kinkade dan Kevin Lim memilih untuk membandingkan antara faktor pemilihan karakter dan gaya bermain tim menggunakan *LR* yang menghasilkan tingkat akurasi sebesar 63% pada faktor pemilihan karakter, dan 73% pada faktor gaya bermain tim [1]. Akan tetapi, metode *LR* tidak memiliki solusi analitik yang dapat digunakan dalam menyelesaikan persamaan linear untuk mendapatkan nilai dari parameter $\beta 0,\ldots,\beta k$, sehingga digunakan metode optimasi numerik untuk menyelesaikan masalah tersebut [3]. Subathra dan Nedunchezhian dalam penelitiannya berhasil meningkatkan akurasi dari LR sebesar 4,98% dengan mengkombinasikannya menggunakan *Particle Swarm Optimization* (*PSO*) pada kasus pengklasifikasian alias [4]. Sedangkan pada penelitian McCaffrey dengan menggunakan *Multi-Swarm Optimization* (*MSO*) yang merupakan variant dari PSO, McCaffrey menyimpulkan bahwa *MSO* cenderung dapat menghasilkan hasil yang lebih optimal dan dapat menangani masalah pengoptimalan lebih baik daripada *PSO* [5].

Berdasarkan uraian tersebut, maka dalam penelitian ini akan digunakan metode *Logistic Regression* dengan optimasi *Multi-Swarm Optimization* untuk memprediksi kemenangan dalam permainan DOTA 2.

1.2 Tujuan

Berdasarkan uraian pada latar belakang yang telah dipaparkan sebelumnya, maksud dari tujuan dalam penelitian ini adalah sebagai berikut :

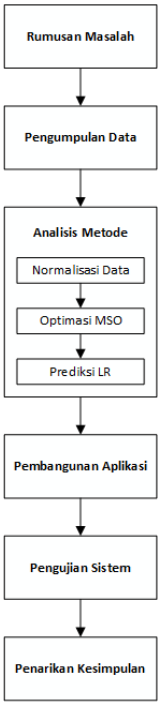
1. Untuk mengimplementasikan algoritma *MSO* yang digunakan untuk mengoptimasi nilai dari parameter *β0,…,βk* dalam metode *LR*.
2. Untuk mengukur nilai akurasi dari penerapan algoritma *LR* dengan optimasi *MSO* dalam memprediksi kemenangan dalam permainan DOTA 2.

2. ISI PENELITIAN

Bagian isi penelitian ini berisi mengenai penjelasan-penjelasan meliputi metode penelitian, *The International*, arsitektur sistem, normalisasi, metode *Multi-Swarm Optimization*, metode *Logistic Regression*, dan hasil pengujian.

2.1 Metode Penelitian

Dalam penelitian ini, terdapat enam alur tahapan yang dilakukan, yaitu dimulai dari tahapan perumusan masalah, tahapan pengumpulan data, tahapan analisis metode yang meliputi normalisasi data, optimasi *MSO*, dan prediksi *LR*, tahapan pembangunan aplikasi, tahapan pengujian sistem dan yang terakhir adalah melakukan penarikan kesimpulan. Berikut merupakan skema metode dalam penelitian ini dapat dilihat pada Gambar 1.



Gambar 1. Skema Metode Penelitian

2.2 *The International*

The International merupakan salah satu turnamen terbesar dalam permanainan DOTA 2 [6]. Setiap hasil pertandingan dari turnamen tersebut akan digunakan sebagai *dataset* dalam penelitian ini. *Dataset* yang digunakan berjumlah sebanyak 1748 data, 409 data dari turnamen *The International 2015*, 398 data dari turnamen *The International 2016*, 540 data dari turnamen *The International 2017*, dan 401 data dari turnamen *The International 2018*. Data-data yang digunakan didapatkan dengan *API* yang disediakan oleh *opendota* [7]. Data tersebut berisi data-data yang didapatkan selama permainan berlangsung. Atribut-atribut yang akan digunakan dapat dilihat pada Tabel 1.

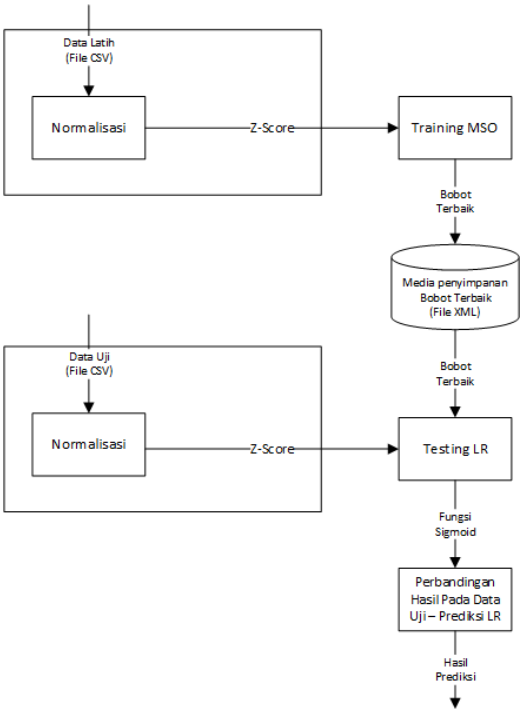
Tabel 1. Atribut pada Data *The International*

Atribut	Keterangan	Atribut	Keterangan
KR	*Kill Radiant*	KD	*Kill Dire*
DR	*Dead Radiant*	DD	*Dead Dire*
AR	*Assist Radiant*	AD	*Assist Dire*
GPMR	*Gold per Minute Radiant*	GPMD	*Gold per Minute Dire*
XPMR	*Experience per Minute Radiant*	XPMD	*Experience per Minute Dire*
LHR	*Last Hit Radiant*	LHD	*Las Hit Dire*
DNR	*Denied Radiant*	DND	*Denied Dire*
GR	*Gold Radiant*	GD	*Gold Dire*

Atribut-atribut yang terdapat pada Tabel 1 kemudian akan direpresentasikan sebagai dimensi yang akan digunakan pada proses optimasi dengan *MSO*, dan prediksi dengan *LR*. Jumlah dimensi yang akan digunakan adalah sebanyak 16 dimensi dengan tambahan satu dimensi, yaitu *β0* yang akan digunakan sebagai pencarian solusi terbaik.

2.3 Arsitektur Sistem

Pembangunan aplikasi yang akan digunakan untuk memprediksi kemenangan dalam permainan DOTA 2 ini meliputi beberapa proses. Proses yang pertama adalah melakukan normalisasi pada data masukan menggunakan *z-score normalization*. Setelah data masukan di normalisasi, kemudian data hasil normalisasi tersebut akan di optimasi pada proses *training MSO*. Setelah mendapatkan solusi yang optimal dari proses *training MSO*, proses yang dilakukan selanjutnya adalah melakukan *testing LR*, pada proses ini prediksi akan dilakukan dengan menggunakan metode *LR*. Untuk keseluruhan proses yang di lakukan dapat di lihat pada blok diagram berikut pada Gambar 2.



Gambar 2. Blok Diagram Sistem yang Dibangun

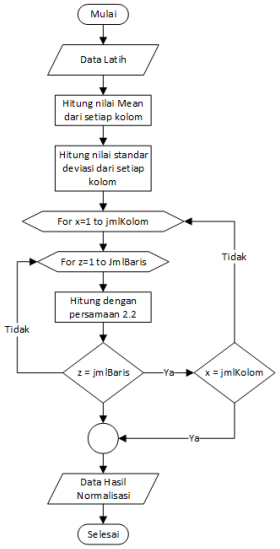
Pada proses normalisasi, data masukan yang digunakan adalah *file* dengan format *CSV*. *File* tersebut berisikan data-data dari hasil pertandingan pada turnamen *The International* dalam permainan DOTA 2. Hasil dari proses normalisasi kemudian akan melalui proses pelatihan dengan menggunakan *MSO*, pelatihan ini bertujuan untuk mendapatkan nilai berupa solusi terbaik (optimal). Solusi optimal yang didapatkan, kemudian akan disimpan ke dalam sebuah *file* dengan format *XML*. *File* yang berisikan bobot terbaik tersebut kemudian akan melalui tahapan pengujian dengan menggunakan *LR*. Pada tahap pengujian ini, data uji yang telah dinormalisasi akan diproses untuk mendapatkan hasil prediksi dengan menggunakan solusi optimal yang telah didapatkan sebelumnya. Hasil dari prediksi yang didapat, kemudian akan dibandingkan dengan hasil aktual pada data yang sebenarnya. Perbandingan dilakukan dengan tujuan untuk mendapatkan nilai akurasi dari metode yang digunakan.

2.4 Normalisasi

Proses pengolahan data yang dilakukan untuk mengubah data awal ke dalam bentuk lain yang bertujuan untuk menjaga rentang/skala pada data agar data yang digunakan lebih tepat untuk dilakukan analisis dinamakan proses normalisasi [8]. Dalam penelitian ini, metode yang digunakan untuk melakukan normalisasi adalah *z-score normalization*. Perhitungan dengan metode *z-score normalization* dapat dilakukan dengan menggunakan persamaan berikut [9]:

$$v' = \frac{v-\bar{x}}{\sigma_x} \quad (1)$$

Terdapat beberapa proses yang dilakukan dalam melakukan normalisasi data. Berikut adalah gambaran alur dari keseluruhan proses yang dilakukan dapat dilihat pada Gambar 3.



Gambar 3. Alur Proses Normalisasi Data

Pada Gambar 3, untuk menghitung nilai dari *z-score* terlebih dahulu harus dicari nilai mean dan nilai standar deviasi dari data. Untuk menghitung nilai mean digunakan persamaan berikut [10]:

$$\bar{x} = \frac{\sum_{i=0}^{n} x_i}{n} \quad (2)$$

Sedangkan untuk menghitung nilai dari standar deviasi digunakan persamaan berikut [10]:

$$\sigma_x = \sqrt{\frac{\sum(x_i-\mu)^2}{N}} \quad (3)$$

Jika penghitungan nilai dari standar deviasi hanya dilakukan pada data sampel (tidak dilakukan pada keseluruhan data), maka dalam melakukan perhitungan digunakan persamaan berikut [10]:

$$s_x = \sqrt{\frac{\sum(x_i-\bar{x})^2}{n-1}} \quad (4)$$

Dalam penelitian ini, proses dari normalisasi data dilakukan dengan tujuan untuk menjaga agar rentang/skala dari setiap dimensi pada *dataset* memiliki bobot yang sama.

2.5 *Multi-Swarm Optimization*

Salah satu metode optimasi dalam *machine learning* yang dapat digunakan untuk memperkirakan sebuah solusi bagi suatu permasalahan numerik yang rumit adalah *Multi-Swarm Optimization* (*MSO*) [11]. Metode *MSO* adalah salah satu variasi yang dikembangkan berdasarkan metode *Particle Swarm Optimization*

(*PSO*) [11]. Pada tahun 1995, untuk pertama kali nya *PSO* diperkenalkan oleh Eberhart dan Kennedy [12]. *MSO* adalah bagian dari *Swarm Intelligence*. *Swarm Intellegence* adalah salah satu teknik kecerdasan buatan yang didasarkan pada perilaku kolektif dan dapat mengatur dirinya sendiri [12].

Terdapat beberapa parameter yang digunakan dalam melakukan optimasi *MSO*. Parameter-parameter yang digunakan dapat dilihat pada Tabel 2 [5].

Tabel 2. Parameter *MSO*

Parameter	Nilai
Swarm	4
Partikel	3
Iterasi Maksimum	5
Batas Minimal Posisi Partikel	-10
Batas Maksimal Posisi Partikel	10
Batas Minimal Kecepatan Partikel	-1
Batas Maksimal Kecepatan Partikel	1
Bobot Inersia (*w)*	0.729
Bobot Kognitif (*c1*)	1.49445
Bobot Sosial (*c2*)	1.49445
Bobot Global (*c3*)	0.3645

Pada penelitian ini, dalam melakukan proses optimasi dengan menggunakan *MSO* digunakan langkah-langkah sebagai berikut [11]:

1. Inisialisasi posisi dan kecepatan partikel secara acak.

   Pada tahap ini, penggunaan jumlah swarm dan partikel akan ditentukan, serta pembeerian nilai pada posisi dan kecepatan partikel secara acak. Untuk melakukan inisialisasi posisi partikel secara acak digunakan persamaan berikut [11]*:*

   $$x_i = minX + r_i(maxX - minX) \qquad (5)$$

   Sedangkan untuk melakukan inisialisasi kecepatan partikel secara acak digunakan persamaan sebagai berikut [11]:

   $$x_i = minX + r_i(maxX - minX) \qquad (6)$$

2. Mengevaluasi optimisasi nilai fungsi pada setiap partikel.

   Evaluasi nilai fungsi pada setiap partikel dilakukan untuk mendapatkan partikel dengan nilai fungsi terbaik. Metode yang digunakan untuk mendapatkan nilai fungsi adalah menggunakan *MSE* (*Mean Squared Error*). Berikut adalah persamaan dari MSE yang digunakan [9]:

   $$MSE = \frac{1}{n}\sum_{i=1}^{n}\left(Y_i - \check{Y}_i\right)^2 \qquad (7)$$

3. Menentukan *bestPartikel.*

   bestPartikel adalah posisi terbaik dari setiap partikel. Posisi ini merupakan posisi yang ditempati oleh partikel saat ini.

4. Menentukan *bestSwarm.*

   bestSwarm adalah posisi terbaik dari setiap swarm. Posisi ini merupakan posisi yang ditempati oleh *swarm* saat ini.

5. Menentukan *bestMulti-Swarm.*

   bestMulti-Swarm adalah posisi terbaik dari setiap *Multi-Swarm*. Posisi ini merupakan posisi yang ditempati oleh *Multi-Swarm* saat ini.

6. Perbarui kecepatan partikel(*velocity*).

   Perhitungan nilai *velocity* dilakukan untuk menentukan posisi baru untuk setiap partikel dengan ketentuan :

   a. Jika v > 1, maka v = 1
   b. Jika -1 <= v <= 1, maka v = v
   c. Jika v < -1, maka v = -1

   Berikut adalah persamaan yang digunakan untuk menentukan nilai *velocity* [5]:

   $$v\,(t+1) = w.v(t) + (c_1.r_1)(p(t) - x(t)) + (c_2.r_2)(s(t) - x(t)) + (c_3.r_3)(g(t) - x(t)) \qquad (8)$$

7. Perbarui posisi partikel.

   Setelah melakukan perhitungan velocity yang baru pada persamaan (8), selanjutnya adalah melakukan perhitungan untuk menentukan posisi partikel yang baru dengan ketentuan :

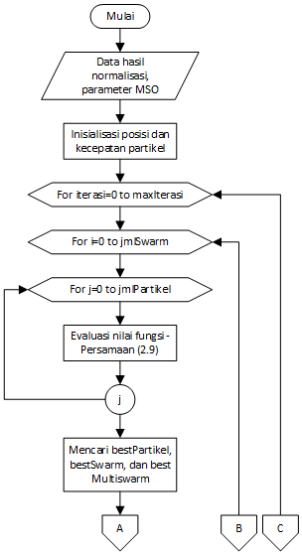
   a. Jika t > 10, maka t = 10
   b. Jika -10 <= t <= 10, maka t = t
   c. Jika t < -10, maka t = -10

   Berikut merupakan persamaan yang digunakan untuk memperbarui posisi partikel [11]:

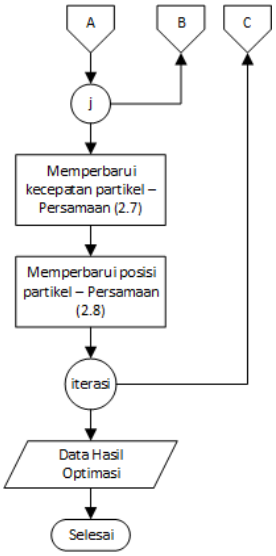
   $$x(t+1) = x(t) + v(t+1) \qquad (9)$$

8. Melakukan perulangan dari langkah ke-2 hingga kriteria terpenuhi, kriteria dikatakan terpenuhi jika telah mendapatkan nilai fungsi yang cukup baik atau apabila telah mencapai batas iterasi maksimum.

Berikut merupakan keseluruhan alur yang dilakukan dalam proses optimasi dengan menggunakan metode *MSO* dapat dilihat pada Gambar 4 dan 5.



Gambar 4. Alur Proses Optimasi *MSO*

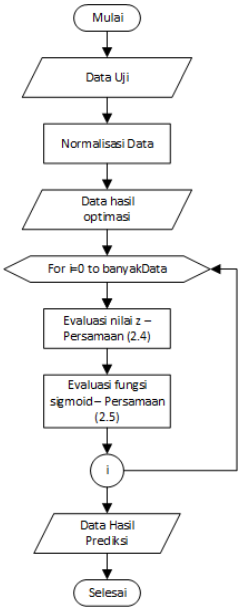


Gambar 5. Alur Proses Optimasi *MSO* - Lanjutan

2.6 *Logistic Regression*

Logistic Regression (*LR*) merupakan salah satu algoritma dalam machine learning yang digunakan untuk melakukan klasifikasi [13]. Metode ini dikembangkan oleh seorang ahli statistik bernama David Cox pada tahun 1958 [13]. Tujuan dari klasifikasi *LR* adalah untuk membuat sebuah model yang dapat menganalisis suatu *dataset* yang memiliki satu atau lebih variabel *independent* yang akan mempengaruhi hasil dari variabel *dependent*, hasil dari variabel *dependent* tersebut dapat berupa 1/0, ya/tidak, dan benar/salah [13].

Dalam penelitian ini *dependent* variabel yang digunakan adalah menang/kalah. Berikut adalah alur dari keseluruhan proses prediksi yang dilakukan dengan menggunakan metode *LR* dapat dilihat pada Gambar 6.



Gambar 6. Alur Proses Prediksi *LR*

Pada Gambar 6, *dataset* yang telah melalui tahap normalisasi dan optimasi, *dataset* tersebut selanjutnya akan masuk ke tahapan untuk mengevaluasi persamaan linear nilai *z* dan fungsi sigmoid. Perhitungan dalam mengevaluasi persamaan linear dari nilai z dapat dihitung dengan menggunakan persamaan [13]:

$$z = \alpha + \beta 1X1 + \ldots + \beta kXk \qquad (10)$$

Sedangkan untuk perhitungan nilai fungsi sigmoid dapat dilakukan dengan menggunakan persamaan [13]:

$$f(z) = 1/(1 + e\hat{}(-z)) \qquad (11)$$

Hasil dari perhitungan fungsi sigmoid dengan menggunakan persamaan (11), akan menghasilkan nilai dengan rentang (0, 1). Hasil tersebut kemudian akan diklasifikasikan menjadi dua kelas, yaitu menang dan kalah dengan ketentuan :

a. Jika *f*(*z*) <= 0.5, maka *f*(*z*) = *false* (kalah)
b. Jika *f*(*z*) >= 0.5, maka *f*(*z*) = *true* (menang)

Hasil dari prediksi tersebut kemudian akan dibandingkan dengan hasil aktual pada *dataset* untuk mengetahui nilai akurasi yang didapatkan dari metode yang digunakan.

2.7 Hasil Pengujian

Pengujian yang dilakukan melalui lima tahapan yaitu pengujian jumlah *swarm*, pengujian jumlah *partikel*, pengujian jumlah iterasi, pengujian batas posisi, dan pengujian batas kecepatan. Masing-masing pengujian akan dilakukan sebanyak 10 kali dan akan diambil nilai rata-rata terendah dari keseluruhan pengujian yang telah dilakukan. Setelah didapatkan nilai terbaik dari semua parameter, kemudian akan dilakukan prediksi pada *dataset* dengan menggunakan parameter terbaik.

2.7.1 Pengujian Jumlah *Swarm*

Pengujian jumlah *swarm* dilakukan dengan tujuan untuk mengetahui jumlah *swarm* yang perlu digunakan untuk mendapatkan nilai fungsi dan bobot yang optimal (terbaik). Nilai fungsi disini merupakan nilai kesalahan atau nilai *error*, dimana nilai kesalahan terbaik adalah nilai terkecil. Jumlah *swarm* yang digunakan pada pengujian ini adalah 1, 2, 3, 4, dan 5. Hasil dari pengujian jumlah *swarm* yang telah dilakukan dapat dilihat pada Tabel 3.

Tabel 3. Pengujian Jumlah Swarm

Jumlah Swarm	Nilai Fungsi Terbaik Percobaan Ke-i										Rata-Rata Nilai Fungsi
	1	2	3	4	5	6	7	8	9	10	
1	0.0643	0.2788	0.2628	0.1439	0.0424	0.1478	0.5500	0.0304	0.2103	0.2958	0.2026
2	0.2018	0.4288	0.0384	0.0530	0.0445	0.0526	0.0298	0.1238	0.0399	0.0767	0.1089
3	0.0510	0.0224	0.0395	0.0547	0.0392	0.0237	0.0323	0.0696	0.0344	0.0304	0.0397
4	0.0371	0.0333	0.0239	0.1164	0.0347	0.0172	0.0307	0.0250	0.0729	0.0592	0.0450
5	0.0289	0.0260	0.0402	0.0423	0.0534	0.0412	0.0341	0.0267	0.1248	0.0588	0.0476

2.7.2 Pengujian Jumlah Partikel

Pengujian jumlah partikel dilakukan dengan tujuan untuk mengetahui jumlah partikel yang perlu digunakan untuk mendapatkan nilai fungsi dan bobot yang optimal (terbaik). Nilai fungsi disini merupakan nilai kesalahan atau nilai *error*, dimana nilai kesalahan terbaik adalah nilai terkecil. Jumlah partikel yang digunakan pada pengujian ini adalah 3, 6, 9, 12, dan 15. Hasil dari pengujian jumlah partikel yang telah dilakukan dapat dilihat pada Tabel 4.

Tabel 4. Pengujian Jumlah Partikel

Jumlah Partikel	Nilai Fungsi Terbaik Percobaan Ke-i										Rata-Rata Nilai Fungsi
	1	2	3	4	5	6	7	8	9	10	
3	0.0383	0.0568	0.1614	0.0496	0.0791	0.0411	0.1309	0.0756	0.1285	0.0425	0.0804
6	0.0267	0.0413	0.0552	0.0320	0.0234	0.0402	0.0320	0.0212	0.0186	0.0219	0.0312
9	0.0226	0.0252	0.0310	0.0287	0.0262	0.0448	0.0190	0.0380	0.0190	0.0277	0.0282
12	0.0233	0.0243	0.0292	0.0258	0.0212	0.0238	0.0234	0.0230	0.0303	0.0236	0.0248
15	0.0237	0.0145	0.0225	0.0242	0.0203	0.0310	0.0184	0.0195	0.0247	0.0296	0.0228

2.7.3 Pengujian Jumlah Iterasi

Pengujian jumlah iterasi dilakukan dengan tujuan untuk mengetahui jumlah iterasi yang perlu digunakan untuk mendapatkan nilai fungsi dan bobot yang optimal (terbaik). Nilai fungsi disini merupakan nilai kesalahan atau nilai *error*, dimana nilai kesalahan terbaik adalah nilai terkecil. Jumlah iterasi yang digunakan pada pengujian ini adalah 10, 20, 30, 40, dan 50. Hasil dari pengujian jumlah iterasi yang telah dilakukan dapat dilihat pada Tabel 5.

Tabel 5. Pengujian Jumlah Iterasi

Jumlah Iterasi	Nilai Fungsi Terbaik Percobaan Ke-i										Rata-Rata Nilai Fungsi
	1	2	3	4	5	6	7	8	9	10	
10	0.0202	0.0302	0.0169	0.0188	0.0196	0.01602	0.01545	0.02813	0.03466	0.01872	0.02187
20	0.0107	0.0100	0.0170	0.0144	0.0129	0.01256	0.01064	0.01197	0.01202	0.00947	0.01216
30	0.0097	0.0098	0.0106	0.0094	0.0121	0.00984	0.00948	0.00993	0.00912	0.01035	0.01004
40	0.0097	0.0086	0.0084	0.0086	0.0088	0.00916	0.00886	0.00884	0.00957	0.01027	0.00907
50	0.0093	0.0085	0.0089	0.0093	0.0095	0.00906	0.00832	0.00970	0.01004	0.00850	0.00911

2.7.4 Pengujian Batas Posisi

Pengujian batas posisi dilakukan dengan tujuan untuk mengetahui batas posisi yang perlu digunakan untuk mendapatkan nilai fungsi dan bobot yang optimal (terbaik). Nilai fungsi disini merupakan nilai kesalahan atau nilai *error*, dimana nilai kesalahan terbaik adalah nilai terkecil. Batas posisi yang digunakan pada pengujian ini adalah (-5, 5), (-10, 10), (-15,15), (-20, 20), dan (-25, 25). Hasil dari pengujian batas posisi yang telah dilakukan dapat dilihat pada Tabel 6.

Tabel 6. Pengujian Batas Posisi

P.Min	P.max	Nilai Fungsi Terbaik Percobaan Ke-i										Rata-Rata Nilai Fungsi
		1	2	3	4	5	6	7	8	9	10	
-5	5	0.0108	0.0100	0.0103	0.0094	0.0097	0.0090	0.0093	0.0101	0.0094	0.0097	0.0098
-10	10	0.0088	0.0086	0.0087	0.0093	0.0089	0.0108	0.0097	0.0121	0.0095	0.0095	0.0096
-15	15	0.0091	0.0094	0.0113	0.0085	0.0083	0.0093	0.0129	0.0106	0.0115	0.0094	0.0100
-20	20	0.0099	0.0125	0.0099	0.0105	0.0117	0.0151	0.0114	0.0088	0.0104	0.0116	0.0112
-25	25	0.0175	0.0115	0.0272	0.0105	0.0157	0.0215	0.0100	0.0150	0.0097	0.0127	0.0151

2.7.5 Pengujian Batas Kecepatan

Pengujian batas kecepatan dilakukan dengan tujuan untuk mengetahui batas kecepatan yang perlu digunakan untuk mendapatkan nilai fungsi dan bobot yang optimal (terbaik). Nilai fungsi disini

merupakan nilai kesalahan atau nilai *error*, dimana nilai kesalahan terbaik adalah nilai terkecil. Batas kecepatan yang digunakan pada pengujian ini adalah (-1, 1), (-2, 2), (-3, 3), (-4, 4), dan (-5, 5). Hasil dari pengujian batas kecepatan yang telah dilakukan dapat dilihat pada Tabel 7.

Tabel 7. Pengujian Batas Kecepatan

K.Min	K.max	Nilai Fungsi Terbaik Percobaan Ke-i										Rata-Rata Nilai Fungsi
		1	2	3	4	5	6	7	8	9	10	
-1	1	0.0086	0.0099	0.0107	0.0089	0.0087	0.0087	0.0092	0.0087	0.0089	0.0091	0.0091
-2	2	0.0096	0.0090	0.0085	0.0098	0.0092	0.0094	0.0089	0.0091	0.0120	0.0097	0.0095
-3	3	0.0091	0.0088	0.0095	0.0100	0.0097	0.0093	0.0093	0.0093	0.0112	0.0098	0.0096
-4	4	0.0097	0.0096	0.0169	0.0102	0.0152	0.0107	0.0145	0.0099	0.0102	0.0089	0.0116
-5	5	0.0132	0.0134	0.0138	0.0096	0.0108	0.0122	0.0091	0.0115	0.0097	0.0108	0.0114

2.7.6 Kesimpulan Pengujian Metode

Dari keseluruhan pengujian yang dilakukan, berikut adalah parameter-parameter terbaik yang didapatkan untuk menghasilkan nilai fungsi dan bobot yang optimal. Berikut adalah parameter terbaik yang didapatkan selama proses pengujian :

a. Batas posisi partikel : (-10, 10)
b. Batas kecepatan partikel : (-1, 1)
c. Jumlah swarm : 3
d. Jumlah partikel : 15
e. Jumlah iterasi maksimum : 40
f. Konst bobot inersia (*w*) : 0,729
g. Konst bobot kognitif (*c1*) : 1,49445
h. Konst bobot sosial (*c2*) : 1,49445
i. Konst bobot global (*c3*) : 0,3645

Inisialisasi posisi dan kecepatan partikel yang dilakukan secara acak akan menghasilkan hasil optimasi yang berbeda-beda walaupun dengan menggunakan parameter yang sama. Oleh karena itu, dengan menggunakan parameter terbaik yang didapatkan dari pengujian, kemudian akan dilakukan optimasi sebanyak 10 kali dengan menggunakan parameter terbaik yang didapatkan. Hasil dari optimasi yang dilakukan dapat dilihat pada Tabel 8.

Tabel 8. Hasil Optimasi

	Nilai Fungsi Terbaik Percobaan Ke-i									
	1	2	3	4	5	6	7	8	9	10
Nilai Fungsi	0.0101	0.0083	0.0092	0.0095	0.0092	0.0092	0.0132	0.0105	0.0093	0.0087
Akurasi	96.01%	96.01%	96.01%	97.01%	95.76%	96.26%	95.76%	97.26%	97.51%	96.51%

Dari hasil optimasi yang didapatkan melalui 10 kali pengujian yang dilakukan, didapatkan nilai akurasi terendah sebesar 95,76% dan nilai akurasi tertinggi sebesar 97,51%. Berikut adalah bobot terbaik yang didapatkan pada nilai akurasi terendah dapat dilihat pada Tabel 9 dan 10.

Tabel 9. Bobot Terbaik Hasil Optimasi Terendah

b0	KR	DR	AR	GPMR	XPMR	LHR	DNR	GR
2.8343	1.3366	-5.7329	-0.8344	8.4690	3.2384	-4.8872	-0.2276	1.0626

Tabel 10. Bobot Terbaik Hasil Optimasi Terendah - Lanjutan

KD	DD	AD	GPMD	XPMD	LHD	DND	GD
2.8031	2.0400	2.0813	-9.1808	-3.3933	3.8595	0.8637	-1.3780

Dengan menggunakan bobot terbaik hasil optimasi terendah pada Tabel 9 dan 10, didapatkan hasil akurasi prediksi kemenangan pertandingan dengan hasil perbandingan sebagai berikut pada Tabel 11.

Tabel 11. Hasil Perbandingan Terendah

Total Data	Total Perbandingan Benar	Total Perbandingan Salah	Akurasi
401	384	17	95.76%

Sedangkan bobot terbaik yang didapatkan pada nilai akurasi tertinggi dapat dilihat pada Tabel 12 dan 13.

Tabel 12. Bobot Terbaik Hasil Optimasi Tertinggi

b0	KR	DR	AR	GPMR	XPMR	LHR	DNR	GR
0.5840	1.1124	-2.6377	0.2268	4.0970	1.2219	-3.2434	-0.1159	4.8480

Tabel 13. Bobot Terbaik Hasil Optimasi Tertinggi - Lanjutan

KD	DD	AD	GPMD	XPMD	LHD	DND	GD
2.8031	2.0400	2.0813	-9.1808	-3.3933	3.8595	0.8637	-1.3780

Dengan menggunakan bobot terbaik hasil optimasi tertinggi pada Tabel 12 dan 13, didapatkan hasil akurasi prediksi kemenangan pertandingan dengan hasil perbandingan sebagai berikut pada Tabel 14.

Tabel 14. Hasil Perbandingan Tertinggi

Total Data	Total Perbandingan Benar	Total Perbandingan Salah	Akurasi
401	391	10	97.51%

Dari perbandingan hasil pada data uji dengan hasil prediksi, dengan menggunakan bobot terbaik hasil optimasi terendah, didapatkan hasil perbandingan yang sama sebanyak 384 data dengan hasil perbandingan yang berbeda sebanyak 17 data dan tingkat akurasi sebesar 95,76%. Sedangkan dengan menggunakan bobot terbaik hasil optimasi tertinggi, didapatkan hasil perbandingan yang sama sebanyak 391 data dengan hasil perbandingan yang

berbeda sebanyak 10 data dan tingkat akurasi sebesar 97,51%. Hasil tersebut menunjukkan bahwa metode yang digunakan dapat menghasilkan solusi yang optimal.

3. PENUTUP

Berdasarkan hasil penelitian yang telah dilakukan, dapat disimpulkan bahwa penerapan metode *Logistic Regression* (*LR*) dengan optimasi *Multi-Swarm Optimization* (*MSO*) dalam memprediksi kemenangan dalam permainan DOTA 2 dalam sistem ini dapat menghasilkan bobot yang sangat baik melalui proses optimasi, bobot tersebut kemudian digunakan dalam proses prediksi pertandingan menggunakan metode *Logistic Regression* dan menghasilkan tingkat akurasi terendah sebesar 95,76% dengan akurasi tertinggi sebesar 97,51%.

Penelitian ini dapat dikembangkan dengan penambahan jumlah fitur yang mempengaruhi proses prediksi, salah satunya adalah item yang sangat mempengaruhi kondisi berjalannya permainan selama permainan berlangsung. Penambahkan beberapa fitur seperti fitur prediksi berdasarkan pemilihan karakter dalam permainan, ataupun berdasarkan para pemain yang memainkan permainan DOTA 2 juga dapat dilakukan, berhubung akses data diberbagai macam pertandingan yang mudah didapatkan.

DAFTAR PUSTAKA

[1] N. Kinkade dan K. Lim, "DOTA 2 Win Prediction", Technical Report, University of California, San Diego, 2015.

[2] Z. Li, D. Cui, dan C. Li, "Dota2 Outcome Prediction", Technical Report, University of California, San Diego, 2017.

[3] V. Robles, C. Bielza, P. Larranaga, S. Gonzalez, dan L. Ohno-Machado, "Optimizing logistic regression coefficients for discrimination and calibration using estimation of distribution algorithms", *TOP - An Official Journal of the Spanish Society of Statistics and Operations Research*, vol. 16, h. 345, 2008.

[4] M. Subathra dan R. Nedunchezhian, "An Improved Alias Classification using Logistic Regression with Particle Swarm Optimization", *Indian J. Sci. Technol.*, vol. 8, no. 28, h. 1–5, 2016.

[5] J. McCaffrey, "Test Run - Logistic Regression Classification with Multi-Swarm Optimization", *Microsoft Developer Network*, Jan-2015.

[6] Valve Corporation, *The International - DOTA 2 CHAMPIONSHIP*, http://www.dota2.com/international/overview/, 15 September 2018 14.57

[7] A. Cui, H. Chung, dan N. Hanson-Holtry, *Open Source DOTA 2 Data Platform*, https://www.opendota.com/, 15 September 2018 15.20

[8] C. C. Aggarwal, *Data Mining - The Textbook*. New York: Springer, 2015.

[9] J. Han dan M. Kamber, *Data Mining: Concepts and Techniques*, 2 ed. San Francisco: Morgan Kauffman, 2006.

[10] R. A. Gordon, Applied Statistics for the Social and Health Sciences. New York: Routledge, 2012.

[11] J. McCaffrey, 'Test Run - Multi-Swarm Optimization', *Microsoft Developer Network*, Feb-2013.

[12] G. Hermawan, "Implementasi Algoritma *Particle Swarm Optimization* Untuk Penentuan Posisi Strategis *Agent* Pada Simulasi Robot Sepak Bola Dua Dimensi", Jurnal Ilmiah Komputer dan Informatika (KOMPUTA), pp. 63-70, 2012.

[13] D. G. Kleinbaum dan M. Klein, *Logistic Regression - A Self-Learning Text*, 3 ed. New York: Springer, 2010.