Menentukan Aksi Lawan Komputer Pada *Game* Strategi Menggunakan Algoritma *K-Nearest Neighbour*

http://dx.doi.org/10.28932/jutisi.v8i3.5137

Riwayat Artikel
Received: 21 Juli 2022 | *Final Revision*: 08 Desember 2022 | *Accepted*: 08 Desember 2022

Michael Freddy [#1], Teady Matius Surya Mulyana [✉#2]

[#1,2] *Program Studi Informatika, Universitas Bunda Mulia*
Jalan Lodan Raya No.2, Ancol, Jakarta Utara 14430
[1]s32180033@student.ubm.ac.id
[2]tmulyana@bundamulia.ac.id
✉Corresponding author: tmulyana@bundamulia.ubm.ac.id

Abstrak — **Kemajuan teknologi komputer memperbolehkan berbagai perangkat untuk menyelesaikan komputasi rumit terutama pada bidang industri hiburan dan contohnya adalah *game*. *Game* strategi merupakan jenis *game* yang paling sering diimplementasikan sistem *Artificial Intelligence* atau AI untuk menirukan tingkah laku manusia saat bermain *game*. Sistem AI game banyak yang mudah ditebak sehingga pemain cepat bosan, maka dibutuhkan AI adaptif dan sederhana untuk memudahkan pengembang *game*. *K-Nearest Neighbour* merupakan algoritma klasifikasi dengan *supervised learning*, algoritma ini akan digunakan dalam penelitian ini. Metode penelitian menguji tingkat akurasi algoritma dalam menentukan kelas aksi dengan memberikan data sampel yang dibagi menjadi data *training* dan data uji. Pengukuran tingkat akurasi dihitung menggunakan *confusion matrix* setelah didapat tabel pengujian. Hasil penelitian menyimpulkan bahwa algoritma *K-Nearest Neighbour* dapat menentukan aksi lawan komputer dengan baik. Dibutuhkan lebih banyak lagi data sampel sebagai data *training* untuk meningkatkan tingkat akurasi klasifikasi.**

Kata kunci— ***Artificial Intelligence*; *Confusion Matrix*; *Strategy Game; K-Nearest Neighbour*.**

Determining Computer Opponent's Actions in Strategy Game Using K-Nearest Neighbour Algorithm

Abstract — ***Advances in computer technology allow various devices to complete complex computing, especially in the entertainment industry and the biggest example is games. Strategy game is the type of game that most often gets an Artificial Intelligence or AI system implemented to imitate human behaviour when playing games. Many game AI systems are predictable so players get bored quickly, so adaptive and simple AI is needed to make it easier for game developers. K-Nearest Neighbour is a classification algorithm with supervised learning, this algorithm will be used in this study. The research method tests the level of accuracy in determining the class by providing a sample of data which is divided into training data and test data. The measure of the level of accuracy is calculated using the confusion matrix after the test table is obtained. The results of the study concluded that the K-Nearest Neighbour algorithm can determine computer opponents fairly well. More data samples are needed as data training to increase the level of classification accuracy.***

Keywords— ***Artificial Intelligence; Confusion Matrix; Strategy Game; K-Nearest Neighbour*.**

I. PENDAHULUAN

Pada jaman kini, perkembangan teknologi komputer kini terbagi menjadi beberapa generasi. Di setiap generasi ada peningkatan yang stabil dalam algoritma, kekuatan komputasi pada sistem komputer, dan bahasa pemrograman. Kemajuan teknologi ini memperbolehkan manusia untuk mempercepat segala macam kegiatan dalam berbagai bidang, dari pembuatan konten, proses kerja dan interaksi fisik dapat dilakukan melalui proses digital. Salah satu bidang yang sangat terbantu dengan adanya teknologi komputer adalah bidang hiburan dan contoh terbesarnya adalah *game*. Pengertian *game* artinya permainan dalam bahasa Inggris, maka *game* komputer adalah permainan elektronik yang dimainkan menggunakan perangkat elektronik dan contohnya komputer. *Game* terdiri dari berbagai variasi atau yang biasa disebut dengan genre. *Game* dibuat oleh seorang atau tim pengembang melalui suatu perancangan dan tujuan yang ditetapkan. Salah satu elemen yang penting dalam pengembangan *game* adalah pemberian tantangan yang terukur dan adil sesuai tingkat kesulitan [1]. Maka demikian, implementasi kecerdasan buatan atau yang biasa dikenal dengan AI (*Artificial Intelligence*) ke dalam *game* merupakan upaya pengembang *game* untuk memberikan sensasi tantangan tersebut.

Pengimplementasian AI ke dalam sebuah *game* sudah sering dijumpai dalam berbagai jenis atau genre permainan seperti permainan tembakan, strategi, hingga balapan. Sistem AI mencoba untuk meniru karakteristik dari proses berpikir manusia untuk membuat sendiri keputusan berpikir tanpa campur tangan manusia. *Game* sudah lama menjadi saran percobaan banyak sistem AI oleh sebagian banyak peneliti yang telah mempelajari algoritma dan teknik pendekatan permainan yang optimal di *game* komputer yang berbeda seperti pada game *Real Time Strategy* (RTS) [2].

Pada saat ini, pengembangan *game* yang menghadirkan AI tanpa kemampuan adaptif sudah banyak dilakukan khususnya pada *game* jenis strategi. Banyak pengembang *game* yang mendapatkan beberapa *feedback* dari pemainnya bahwa *game* nya terlalu membosankan karena mudah ditebak lawan komputernya. Maka itu, sudah banyak pengembang *game* yang mengimplementasikan AI yang dapat belajar dengan sendirinya. AI yang digunakan biasanya memiliki kemampuan untuk mempelajari cara bermain seiring waktu berjalan atau dengan cara mendapatkan data pelatihan yang biasa disebut dengan *supervised learning* [3]. Dengan AI adaptif, pengalaman bermain para pemain dapat dipersonalisasikan sesuai sesi *gameplay* mereka [4].

Bagi beberapa pengembang, membuat *game* dengan AI adaptif terasa rumit. Hal tersebut dikarenakan banyaknya perubahan situasi dan pilihan aksi yang luas, faktor percabangan yang ekstrem, dan pilihan aksi yang mengubah situasi jauh ke masa depan dimana semua faktor tersebut menghambat teknik AI [5]. AI berfungsi berdasarkan algoritma yang dipakai, sehingga tidak akan bertingkah jauh dari algoritma yang dipakai. Karena hal tersebut, dibutuhkan algoritma AI sederhana yang dapat belajar sendiri untuk menyesuaikan performa pemain.

Beberapa peneliti sebelumnya melakukan penelitian dalam percobaan pembuatan *game* menggunakan sebuah metode yang dapat menentukan pemilihan strategi AI nya. Metode yang digunakan pada penelitian tersebut adalah *decision tree* UCB1 sebagai dasar metode perpindahan adaptif level kesulitan pada lawan musuh. Hasilnya, metode tersebut dapat membuat AI dalam *game* beradaptasi dengan performa dari pemainnya dengan cukup baik namun dibutuhkan adanya beberapa modifikasi supaya dapat bekerja dengan optimal [6].

Peneliti lainnya menggunakan logika *Fuzzy* Tsukamoto sebagai dasar metode implementasi AI pada karakter komputer untuk menentukan aksinya sendiri. Implementasi tersebut digunakan pada sebuah *game* dengan genre RPG dimana rata-rata *game* dengan genre tersebut yang beredar di pasaran, masih banyak sistem kecerdasan buatan yang dipakai belum dapat menentukan pilihan keputusan dengan baik. Pada penelitian tersebut, logika *fuzzy* digunakan sebagai dasar metode pembuatan sistem kecerdasan buatan untuk melakukan pembobotan atas pemilihan aksi yang kemungkinan dipilih. Hasil yang didapat pada penelitian tersebut dapat membuat kesimpulan bahwa kemanjuran dari implementasi logika *fuzzy* Tsukamoto sebagai dasar metode pembuatan sistem kecerdasan buatan sebuah game RPG dimana karakter dalam *game* dapat menentukan aksi yang harus dipilih terhadap kondisi yang dihadapi [7].

Salah satu algoritma AI yang dapat digunakan untuk membuat AI adaptif dan sederhana adalah algoritma *K-Nearest Neighbour* yang disingkat menjadi K-NN. Algoritma ini merupakan algoritma *machine learning* dengan pendekatan *supervised learning* yang bekerja untuk melakukan klasifikasi [8]. Metode klasifikasi dipakai dalam sebuah proses untuk menemukan model atau fungsi yang menjelaskan dan mengkarakterisasi konsep atau kelas data, untuk tujuan tertentu. Data *training* diproyeksikan ke dalam ruang basis data yang besar, di mana setiap dimensi mewakili fitur data, yang dibagi menjadi beberapa bagian berdasarkan pembelajaran *instance-based* atau *lazy learning* di mana fungsinya hanya diperkirakan secara lokal dan semua komputasi ditahan hingga klasifikasi [9]. Algoritma K-NN dapat digunakan untuk melakukan klasifikasi dalam beberapa macam hal misalnya untuk klasifikasi produk dari berbagai *e-marketplace* [10] ataupun untuk klasifikasi wajah hewan mamalia [11].

Meskipun K-NN tertuju untuk melakukan klasifikasi, algoritma ini juga bisa saja diimplementasikan ke dalam sebuah *game*. Algoritma K-NN lebih baik digunakan pada penelitian ini daripada logika yang sederhana contohnya logika *IF*. Hal tersebut dikarenakan K-NN dapat melakukan klasifikasi berdasarkan kemiripan pada data sehingga jika dibandingkan dengan logika *IF*, lawan komputer yang memakai K-NN dapat memilih aksi yang lebih masuk akal daripada lawan komputer yang hanya memakai logika *IF*.

Maka dari itu, penelitian ini bertujuan untuk mencari tahu tingkat akurasi penentuan aksi komputer pada *game* strategi yang menggunakan algoritma K-NN. Hasil dari penelitian ini diharapkan dapat memperoleh manfaat untuk mencari tahu seberapa efektif algoritma K-NN jika digunakan dalam sebuah *game* strategi dan manfaat lainnya berupa menjadi referensi kepada peneliti lainnya yang akan mencoba menerapkan sistem AI adaptif ke dalam sebuah *game*.

II. METODE PENELITIAN

A. Prosedur Penelitian

Penelitian ini dilakukan dalam beberapa langkah yang sistematis dan teratur. Langkah-langkah tersebut secara berurut dimulai dari studi literatur, analisis dan perancangan sistem, implementasi sistem, dan pengujian sistem.

1) Studi Literatur: Pada tahapan ini, ditujukan untuk mengumpulkan, mencari, dan mempelajari data referensi yang ditemukan dari sumber-sumber seperti buku, skripsi, jurnal, internet, dan sumber lainnya.

2) Analisis dan Perancangan Sistem: Pada tahap ini, peneliti mengolah data yang telah dikumpulkan dan melakukan analisis terhadap data tersebut sehingga tujuan dalam penelitian ini dapat dicapai menggunakan metode yang disarankan.

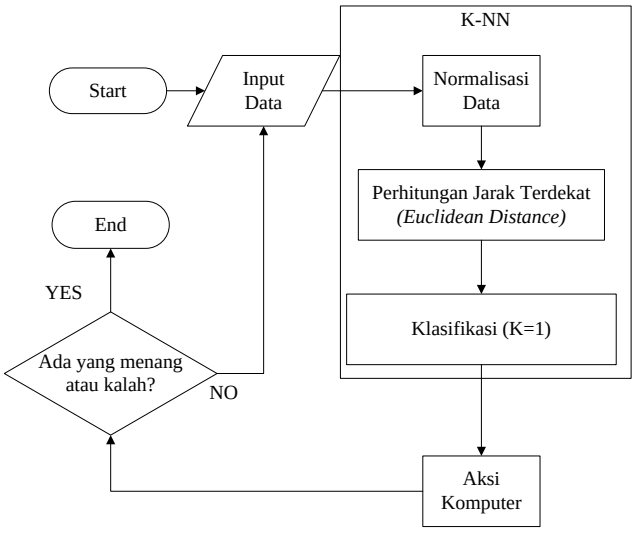
3) Implementasi Sistem: Pada tahap ini, peneliti melakukan pengimplementasian sistem yang dapat menyelesaikan masalah dari penelitian ini.

4) Pengujian Sistem: Tahap ini digunakan untuk melakukan pengujian terhadap algoritma *K-Nearest Neighbour* yang telah dirancang.

B. Rancangan Penelitian

Pada penelitian ini, algoritma yang akan digunakan yaitu *K-Nearest Neighbour* akan diuji ketepatannya dalam menentukan aksi lawan komputer pada *game* strategi menggunakan *confusion matrix*. Matriks tersebut biasanya digunakan dalam evaluasi atau pengukuran sebuah metode klasifikasi. Pada matriks ini, setiap barisnya merepresentasikan suatu kelas aktual dan setiap kolomnya merepresentasikan suatu kelas yang diprediksi dari hasil klasifikasi [12].

Sebelum melakukan implementasi algoritma, dibutuhkan sarana utama untuk pengumpulan data yaitu pembuatan *game*. Setelah *game* selesai dibuat, proses implementasi algoritma secara bertahap dimulai dari penginputan data yang berupa pilihan aksi pemain dan dilanjutkan dengan proses normalisasi data. Tahap berikutnya melakukan perhitungan pada metode K-NN yang merupakan penghitungan jarak terdekat menggunakan rumus *Euclidean* seperti yang tercantum pada rumus (2). Proses dilanjutkan dengan tahap klasifikasi sebagai penentu aksi berdasarkan kelas yang terpilih. *Flowchart* yang menggambarkan keseluruhan proses algoritma pada penelitian dapat dilihat pada Gambar 1.



Gambar 1. *Flowchart* Proses Implementasi Algoritma K-NN

1) Pembuatan Game dan Pengumpulan Data Sampel: Untuk memulai perolehan data, *game* yang menjadi sarana utama mendapatkan data harus dibuat. Data dikumpulkan secara manual dan merupakan data dari dalam permainan antara dua pemain yang berperan sebagai pakar dari dataset yang akan dikumpul ke dalam *file* .csv. *Game* berjenis strategi ini akan dimainkan antara satu lawan satu, jumlah atribut yang sama, dan dengan pilihan aksi sebanyak 3 macam aksi yaitu Menyerang, Bertahan, dan Memulihkan yang masing-masing memiliki label aksi berurut dari A, D, dan R. Aksi-aksi tersebut akan mempengaruhi nilai-nilai atribut, dimana relasinya dijelaskan pada Tabel 1.

TABEL 1
RELASI PENGARUH AKSI TERHADAP NILAI ATRIBUT

Aksi	Atribut Terpengaruh				
	Nyawa (HP)	Nyawa Lawan (HP)	Material (ENG)	Kekuatan (STR)	Pertahanan (DEF)
Menyerang (A)	Tidak Berubah	Berkurang sesuai nilai atribut kekuatan	Dikurangi 20 poin	Dikurangi 1 poin	Bertambah 1 poin
Bertahan (D)	Nilai atribut kekuatan pada lawan dikurangi sesuai dengan nilai atribut pertahanan	Tidak berubah	Tetap berkurang 10 poin, gagal atau berhasil	Bertambah 2 poin jika berhasil menangkis	Dikurangi 1 poin jika berhasil menangkis
Memulihkan (R)	Tidak berubah	Tidak berubah	Bertambah 15 poin	Bertambah 1 poin	Tidak berubah

Setiap aksi dilakukan oleh markasnya milik pemain akan dipengaruhi atribut dari markas masing-masing. Atribut yang mempengaruhi berupa nyawa(HP), material(ENG), kekuatan(STR), dan pertahanan(DEF). Nilai-nilai atribut akan berubah setelah melakukan aksi-aksi tertentu. Jika pemain melakukan aksi Menyerang(A atau *Attack*) dimana aksi mengurangi nyawa musuh sebanyak nilai atribut Kekuatan, kemudian Material akan berkurang 20 poin, Kekuatan berkurang 1 poin, dan Pertahanan bertambah 1 poin. Sedangkan untuk aksi Bertahan (D atau *Defend*), pemain yang melakukan aksi akan menangkis serangan yang diterima, kekuatan serangan yang diterima pemain akan dikurangi dengan nilai atribut Pertahanan, kemudian Material akan berkurang 10 poin, Kekuatan bertambah 2 poin, dan Pertahanan berkurang 1 poin. Kemudian untuk aksi Memulihkan(R atau *Restock)*, nilai atribut yang berubah adalah nilai atribut Material yang bertambah 15 poin dan nilai atribut Kekuatan yang bertambah 1 poin. Penggunaan atribut Material digunakan supaya permainan tidak hanya serba menebak antara ingin menyerang atau bertahan. Jika nilai atribut Material dibawah 20, maka aksi Menyerang tidak dapat dilakukan. Jika nilai atribut Material dibawah 10, maka aksi Menyerang dan Bertahan tidak dapat dilakukan. Kemudian, nilai atribut Material juga dibatasi dengan nilai tertinggi yaitu 100. Maka itu, jika nilai atribut Material lebih dari 85, maka aksi Memulihkan tidak dapat dilakukan.

Sebagai parameter menang atau kalah dalam permainan ini, ditentukan jika atribut nyawa salah satu pemain mencapai 0 dimana pemain dengan atribut nyawa 0 dinyatakan sebagai pemain yang kalah dan pemain lainnya dinyatakan sebagai pemenang. Setelah *game* dapat dimainkan, maka peneliti telah mendapat sarana untuk melakukan pengumpulan data sampel. *Game* dimainkan oleh dua pemain manusia dan setiap kali salah satu pemain melakukan aksi, nilai-nilai atribut dicatat. Setelahnya, kumpulan data dimasukkan ke dalam *file* .csv yang akan digunakan saat implementasi algoritma. Kumpulan data dalam *file* tersebut akan menjadi patokan atau bentuk awal dari dataset dan dijadikan sebagai data *training*. Setelah *game* terbuat dan pengumpulan data sampel selesai, berikutnya masuk ke dalam proses implementasi algoritma *K-Nearest Neighbour*.

2) Proses Algoritma K-Nearest Neighbour: Setelah *game* terbuat dan data sampel terkumpul, berikutnya adalah perancangan proses algoritma *K-Nearest Neighbour*. Algoritma K-NN dirancang supaya dapat menentukan pilihan aksi komputer. Aksi yang ditentukan berdasarkan jarak tetangga data terdekat dengan nilai K=1. Proses menentukan pilihan aksi komputer dilakukan dalam beberapa tahap. Tahapan pertama yaitu melakukan input data.

Pada tahap penginputan data, data yang diinput merupakan kondisi nilai-nilai atribut ketika aksi dilakukan. Aksi yang akan dipilih oleh komputer nantinya berdasarkan kondisi nilai-nilai atribut kelas aksi pada data sampel selama permainan berlangsung. Setelah penginputan data, tahapan berikutnya pada proses adalah melakukan perhitungan yang diawali oleh normalisasi data.

Proses normalisasi ini merubah skala nilai atribut dari data sehingga jarak datanya berada pada *range* tertentu. Proses normalisasi ini akan membuat semua nilai data berubah menjadi dari jarak 0 sampai 1 dengan tujuan untuk memudahkan tahapan selanjutnya. Untuk melakukan normalisasi data, digunakan rumus *min-max* seperti yang ditunjukkan pada Rumus (1).

$$x_{new} = \frac{x_{old} - x_{min}}{x_{max} - x_{min}} \quad (1)$$

Dimana x_{new} adalah nilai data setelah normalisasi, x_{old} adalah nilai data sebelum normalisasi, x_{min} adalah nilai data paling kecil, dan x_{max} adalah nilai data paling besar

Setelah data telah dinormalisasi menjadi pada *range* 0 sampai 1, tahap selanjutnya dalam proses adalah menghitung jarak kedekatan yang akan dihitung menggunakan jarak *Euclidean* yang diperagakan pada Rumus (2). Jarak yang dihitung merupakan jarak antara data *training* dengan data yang akan diuji. Setelah jarak kedekatan dihitung, akan dipilih nilai terkecil yang paling dekat. Nilai yang terpilih akan menjadi penentu pilihan yang akan dipilih lawan komputer dan dianggap sebagai hasil *output*.

$$d_{ij} = \sqrt{\sum_{k=1}^{n}(x_{ik} - x_{jk})^2} \quad (2)$$

Dimana d_{ij} adalah Jarak antara data *training* i dengan data uji j, i adalah indeks data training, j adalah indeks data uji, n adalah jumlah atribut dan x adalah Nilai atribut

Setelah penghitungan jarak pada seluruh data selesai, tahapan selanjutnya adalah proses klasifikasi. Tahap ini dapat dilakukan dengan metode *voting* yang memilih kelas data mayoritas dari sejumlah k data. Nilai k harus ganjil supaya data mayoritas dapat ditentukan. Pada penelitian ini, nilai k adalah 1 sehingga tidak harus melakukan metode *voting*.

Setelah didapatkan hasil *output* berupa kelas yang didapat dari hasil klasifikasi, tahap terakhir pada proses implementasi adalah penentuan aksi. Tahap ini menentukan aksi komputer berdasarkan hasil *output* dari proses klasifikasi. Menggunakan beberapa logika *IF*, aksi komputer akan dipilih berdasarkan kelas yang terpilih pada proses klasifikasi. Proses algoritma K-NN akan dilakukan kembali secara berulang sampai ditemukan pemenang dari permainannya. Data aksi pemain yang menjadi data input pada awal proses implementasi algoritma akan ditambahkan ke dalam dataset dengan tujuan supaya komputer dapat beradaptasi karena mendapat data *training* yang lebih banyak lagi. Dataset yang dimaksud berisikan data yang digunakan sebagai data *training* algoritma K-NN. Isi dari dataset akan direset kembali menjadi bentuk dataset awal jika pengguna atau *user* memilih untuk keluar dari *game*.

Tahap pengujian dilakukan untuk mencari hasil berupa tingkat akurasi algoritma K-NN dalam penentuan aksi lawan komputer pada *game* strategi. Untuk pengujian, dibutuhkan terlebih dahulu kumpulan data sampel yang akan dijadikan sebagai data *training*, kemudian data tersebut diuji untuk dianalisis kebenarannya. Aksi komputer dijadikan sebagai kelas pada penelitian sehingga artinya ada lebih dari 2 kelas atau disebut juga dengan *multi class*. Apabila data sampel sudah sepenuhnya diuji dan terbukti kebenarannya maka artinya data tersebut sudah konvergen. Pengujian untuk mencari tingkat ketepatan atau akurasi akan dilakukan dengan membagi data sampel menjadi data *training* dan data uji.

Sebagai hasil dari pengujian, indikator untuk mengukur performa dari metode klasifikasi K-NN adalah berupa *accuracy*, *precision*, dan *recall*. Rumus-rumus untuk menghitung ketiga indikator tersebut diperagakan pada Rumus (3) untuk *accuracy*, Rumus (4) untuk *precision* dan Rumus (5) untuk *recall*.

$$\boldsymbol{accuracy} = \frac{TP}{TP+TN+FP+FN} \quad (3)$$

$$\boldsymbol{precision} = \frac{TP}{TP+FP} \quad (4)$$

$$\boldsymbol{recall} = \frac{TP}{TP+FN} \quad (5)$$

Dimana TP adalah *True Positive*, TN adalah *True Negative*, FP adalah *False Positive dan FN* adalah *False Negative*

True Positive (TP) merupakan jumlah data klasifikasi yang diprediksi kelas positif pada proses klasifikasi dan sesuai dengan data yang aktual. *True Negative* (TN) merupakan jumlah data klasifikasi yang diprediksi kelas negatif pada proses klasifikasi dan sesuai dengan data yang aktual. *False Positive* (FP) merupakan jumlah data klasifikasi yang diprediksi positif pada proses klasifikasi tetapi tidak sesuai dengan data aktual yang seharusnya diklasifikasi sebagai kelas negatif. *False Negative* (FN) merupakan jumlah data klasifikasi yang diprediksi negatif pada proses klasifikasi tetapi tidak seusai dengan data aktual yang seharusnya diklasifikasi sebagai kelas positif.

Rumus untuk mencari *accuracy* dapat disederhanakan yaitu dengan membagi jumlah data yang hasil klasifikasinya memprediksi kelas aktual dengan jumlah seluruh data. Nilai *precision* dan *recall* akan dicari setelah masing-masing kelas sudah didapatkan nilai *precision* dan *recall* nya dengan menghitung rata-ratanya. Setelah ketiga nilai indikator pengukur tersebut didapat, performa dari proses klasifikasi algoritma dapat disimpulkan.

III. HASIL DAN PEMBAHASAN

Hasil penelitian untuk mencari kesimpulan ditemukan melalui hasil dari pengujian. Hasil pengujian didapat setelah menggunakan sebanyak 200 data sampel yang kemudian dipisah menjadi sebagian data *training* sebanyak 140 data dan sebagian data uji sebanyak 60 data. Algoritma K-NN diuji untuk menentukan aksi lawan komputer berdasarkan 60 data uji tersebut dan ditentukan berdasarkan *training* menggunakan 140 data lainnya. Tabel 2 menunjukkan *confusion matrix* dari pengujian yang dilakukan.

TABEL 2
TABEL *CONFUSION MATRIX MULTI CLASS* PENGUJIAN DATA

Hasil Kelas	Prediksi		
Aktual	Menyerang (A)	Bertahan (D)	Memulihkan (R)
Menyerang (A)	29	0	6
Bertahan (D)	0	8	1
Memulihkan (R)	4	0	12

Dari 60 kali pengujian, jumlah dari proses klasifikasi yang memberikan hasil prediksi yang sesuai dengan hasil aktual adalah sebanyak 49 kali dan jumlah dari proses klasifikasi yang memberikan hasil prediksi yang tidak sesuai dengan hasil aktual adalah sebanyak 11 kali. Berikutnya, menggunakan rumus-rumus yang telah disebut sebelumnya, didapatkan hasil penghitungan indikator pengukur performa yang ditunjukkan pada Tabel 3. Kesalahan pemilihan aksi yang terjadi seperti seharusnya memilih "Memulihkan (R)" malah memilih "Menyerang (A)" yang terjadi sampai 6 kali kekeliruan. Terjadi karena adanya nilai jarak training atribut tersimpan dengan input atribut yang lebih pendek pada "Menyerang (A)". Nilai jarak ini ini disebabkan karena keenam momen nilai-nilai input atribut untuk prediksi "Memulihkan (R)" yang terinput pada sistem memiliki nilai yang yang lebih mendekati dengan pilihan "Menyerang (A)" yang tersimpan pada data training. Demikian juga pada kejadian prediksi "Menyerang (A)" malah terjadi empat kali kejadian aktual "Memulihkan (R)" dengan alasan yang sama. Hal ini terjadi karena dipergunakannya nilai K=1 yang hanya melakukan *voting* dari satu data terpendek saja. Dikarenakan pada penelitian ini fokus penelitian adalah pada kemungkinan mempergunakan KNN saja, maka dipilih K=1 yang lebih sederhana. Nantinya pada penelitian berikutnya akan dilakukan pemilihan berapa nilai K yang lebih layak sehingga mengurangi kesalahan. Dari hasil penghitungan, jumlah proses klasifikasi algoritma K-NN yang menentukan hasil prediksi yang sesuai dengan hasil aktual mencapai 82%. Nilai rata-rata *precision* mencapai 82% dan nilai rata-rata *recall* mencapai 84%.

TABEL 3
TABEL HASIL PENGHITUNGAN PERFORMA METODE

Indikator	Jumlah		
	Menyerang (A)	Bertahan (D)	Memulihkan (R)
TP	29	8	12
FP	6	1	4
FN	4	0	7
Precision (TP/(TP+FP))	83%	89%	75%
Recall (TP/(TP+FN))	88%	100%	63%
Accuracy	((29+8+12) / 60) *100 % = 82%		
Avg. Precision	(83+89+75)/3 = 82%		
Avg. Recall	(88+100+63)/3 = 84%		

Secara teknis, seperti sudah dijelaskan pada sub bab Rancangan Penelitian, algoritma K-NN diterapkan pada penelitian ini dengan melakukan aksi penghitungan pencarian jarak atribut-atribut yang mempengaruhi, berupa nyawa, material, kekuatan, dan pertahanan. Atribut diinput kedalam K-NN setelah pemain melakukan aksi Menyerang (A atau *Attack*), Bertahan (D atau *Defend*), dan Memulihkan (R atau *Restock*) yang mempengaruhi nilai baru dari atribut-atribut tersebut. Input atribut-atribut tersebut dihitung jaraknya mempergunakan nilai *Euclidian Distance* seperti pada rumus (2) terhadap data training tersimpan. Kemudian dengan mempergunakan K-NN dengan nilai K=1, maka diambil nilai jarak terpendek antara input atribut dengan atribut tersimpan. *Record* data terpendek tersebut yang akan dipergunakan kelas aksi nya sebagai kelas aksi terpilih dari *player* komputer. Sebagai contoh proses, dicantumkan data sampel berikut.

TABEL 4
TABEL CONTOH DATA SAMPEL

ID	Nilai Atribut								Kelas Aksi
	HP1	HP2	STR1	STR2	DEF1	DEF2	ENG1	ENG2	
1	200	200	10	10	10	10	100	100	D
2	128	151	8	10	17	16	25	15	A
3	78	108	15	15	20	22	5	60	R
4	25	25	24	17	21	28	0	60	R
5	167	148	9	8	14	15	10	35	R
6	92	78	17	16	22	21	90	15	A
7	25	25	19	26	28	21	90	20	A
8	147	171	9	12	18	6	40	50	?

Disediakan pada tabel 4 sejumlah data dengan data terakhir yang belum diklasifikasi. *ID* data 1 sampai 7 akan digunakan sebagai data *training* untuk mengklasifikasi kelas aksi *ID* data ke-8. Label atribut dengan akhiran 1 menunjukkan atribut pemain yang melakukan aksi (HP1, STR1, ...) sedangkan label atribut dengan akhiran 2 menunjukkan atribut lawannya (HP2, STR2, ...). Untuk mempermudah penghitungan, data disederhanakan menjadi antara *range* 0 sampai 1 menggunakan rumus normalisasi seperti pada rumus (1).

TABEL 5
TABEL CONTOH DATA SAMPEL SETELAH NORMALISASI

ID	Nilai Atribut								Kelas Aksi
	HP1	HP2	STR1	STR2	DEF1	DEF2	ENG1	ENG2	
1	1,00	1,00	0,13	0,11	0,00	0,18	1,00	1,00	D
2	0,59	0,72	0,00	0,11	0,39	0,45	0,25	0,00	A
3	0,30	0,47	0,44	0,39	0,56	0,73	0,05	0,53	R
4	0,00	0,00	1,00	0,50	0,61	1,00	0,00	0,53	R
5	0,81	0,70	0,06	0,00	0,22	0,41	0,10	0,24	R
6	0,38	0,30	0,56	0,44	0,67	0,68	0,90	0,00	A
7	0,00	0,00	0,69	1,00	1,00	0,68	0,90	0,06	A
8	0,70	0,83	0,06	0,22	0,44	0,00	0,40	0,41	?

Setelah data dinormalisasi, berikutnya adalah penghitungan jarak masing-masing nilai atribut dari data *ID* ke-8 dengan setiap data menggunakan rumus (2). Hasil penghitungan jarak ditunjukkan pada tabel 6.

TABEL 6
TABEL CONTOH PENGHITUNGAN JARAK DATA

ID	Jumlah Jarak Semua Nilai Atribut Dengan Data ID Ke-8
1	$\sqrt{(1{,}00-0{,}70)^2+(1{,}00-0{,}83)^2+\cdots}=1{,}04$
2	$\sqrt{(0{,}59-0{,}70)^2+(0{,}72-0{,}83)^2+\cdots}=0{,}67$
3	1,06
4	1,83
5	0,65
6	1,27
7	1,82

ID data yang memiliki jarak terdekat atau terendah dari *ID* data ke-8 adalah *ID* data ke-5 dengan jarak 0,65. Dengan demikian, *ID* data ke-8 dapat diberikan label kelas yang sama dengan label kelas *ID* data ke-5 yaitu R. Aksi kemudian dapat dilakukan sesuai kelas yang didapat.

Hasil pengujian data menunjukkan performa algoritma K-NN yang baik dalam menentukan aksi pada *game*. Aksi aktual merupakan aksi yang disarankan menurut pakar atau pemain manusia, sehingga isi data sampel yang dijadikan data *training* mungkin tidak sepenuhnya benar sebagai aksi yang harus dilakukan. Penentuan aksi oleh algoritma K-NN masih menunjukkan beberapa kekurangan, khususnya pada penentuan aksi dengan kelas label R. Aksi yang ditentukan lainnya juga bisa saja tidak sesuai karena kelas aksi yang terpilih merupakan jarak data yang paling dekat dari data uji. Jarak data didapat berdasarkan penghitungan menggunakan nilai-nilai atribut, sehingga terdapat kemungkinan kelas aksi yang terpilih merupakan kelas aksi yang salah. Hal tersebut dapat terjadi karena setiap nilai atribut pada kelas aksi memiliki prioritas atau bobot yang sama. Selain itu, menambahkan data sampel sebagai data *training* juga akan membantu dalam menentukan kelas aksi yang sesuai.

IV. SIMPULAN

Berdasarkan hasil penelitian yang didapatkan sebelumnya, didapatkan kesimpulan yang diantaranya adalah bahwa algoritma *K-Nearest Neighbour* mampu menentukan aksi lawan komputer pada *game* strategi dengan baik. Data sampel yang digunakan sebanyak 200 data dipisah menjadi 140 data *training* dan sisa 60 data lainnya digunakan sebagai data uji. Pengujian tersebut menghasilkan tingkat akurasi sebesar 82% dalam menentukan kelas aksi yang sesuai dengan kelas aktual. Simpulan lainnya yang didapat adalah dari hasil pengujian, menunjukkan masih adanya kekurangan dalam penentuan kelas aksi karena nilai atribut masing-masing kelas memiliki prioritas atau bobot yang sama. Kemudian data sampel yang dijadikan sebagai data *training* juga harus diperbanyak supaya tingkat akurasi penentuan aksi menjadi lebih akurat lagi.

Sebagai saran untuk melanjutkan ke penelitian berikutnya, peneliti mengusulkan untuk memberikan nilai bobot pada masing-masing atribut supaya penentuan kelas aksi dapat dilakukan lebih baik lagi dan diperlukan juga data sampel yang lebih banyak lagi supaya hasilnya pula menjadi lebih akurat.

UCAPAN TERIMA KASIH

Ucapan terima kasih kepada pembimbing, rekan-rekan dan kawan-kawan lainnya yang sudah membantu peneliti dalam pembuatan penelitian ini.

DAFTAR PUSTAKA

[1] M. Urh, G. Vukovic, E. Jereb and R. Pintar, "The Model for Introduction of Gamification into E-learning in Higher Education," *Procedia - Social and Behavioral Sciences,* vol. 197, p. 388–397, Jul 2015.

[2] D. Perez-Liebana, S. Samothrakis, J. Togelius, T. Schaul and S. Lucas, "General Video Game AI: Competition, Challenges and Opportunities," *AAAI,* vol. 30, no. 1, Mar 2016.

[3] D. Petersson, "Supervised Learning," 26 Mar 2021. [Online]. Available: https://www.techtarget.com/searchenterpriseai/definition/supervised-learning. [Accessed 01 Jun 2022].

[4] F. Foffano and D. Thue, "Changes of user experience in an adaptive game," in *Proceedings of the 14th International Conference on the Foundations of Digital Games,* 2019.

[5] D. Gomme and R. Bartle, "Strategy Games: The Components of A Worthy Opponent," in *International Conference on the Foundations of Digital Games,* 2020.

[6] E. Santoso, G. S. Budhi and R. Intan, "Pembuatan Game Dengan Menerapkan Metode Decision Tree: UCB1, Untuk Menentukan Pemilihan Strategy Dalam AI," *Jurnal Infra,* vol. 5, no. 1, p. 60–66, 2017.

[7] D. Ratanajaya and H. A. Wibawa, "Implementasi Kecerdasan Buatan dalam Menentukan Aksi Karakter pada Game RPG dengan Logika Fuzzy Tsukamoto," *Khazanah Informatika: Jurnal Ilmu Komputer dan Informatika,* vol. 4, no. 2, p. 82, Dec 2018.

[8] S. Hussein, "Mengenal K-Nearest Neighbor: Algoritma Populer untuk Machine Learning," 27 Oct 2021. [Online]. Available: https://geospasialis.com/k-nearest-neighbor/. [Accessed 1 Jun 2022].

[9] A. R. Lubis, M. Lubis and A.-. Khowarizmi, "Optimization of distance formula in K-Nearest Neighbor method," *Bulletin of Electrical Engineering and Informatics,* vol. 9, no. 1, p. 326–338, Feb 2022.

[10] D. Sebastian, "Implementasi Algoritma K-Nearest Neighbor untuk Melakukan Klasifikasi Produk dari beberapa E-marketplace," *Jurnal Teknik Informatika dan Sistem Informasi,* vol. 5, no. 1, May 2019.

[11] Y. Yohannes, Y. P. Sari and I. Feristyani, "Klasifikasi Wajah Hewan Mamalia Tampak Depan Menggunakan k-Nearest Neighbor Dengan Ekstraksi Fitur HOG," *Jurnal Teknik Informatika dan Sistem Informasi,* vol. 5, no. 1, May 2019.

[12] O. Caelen, "A Bayesian interpretation of the confusion matrix," *Annals of Mathematics and Artificial Intelligence,* vol. 81, no. 3–4, p. 429–450, Dec 2017.