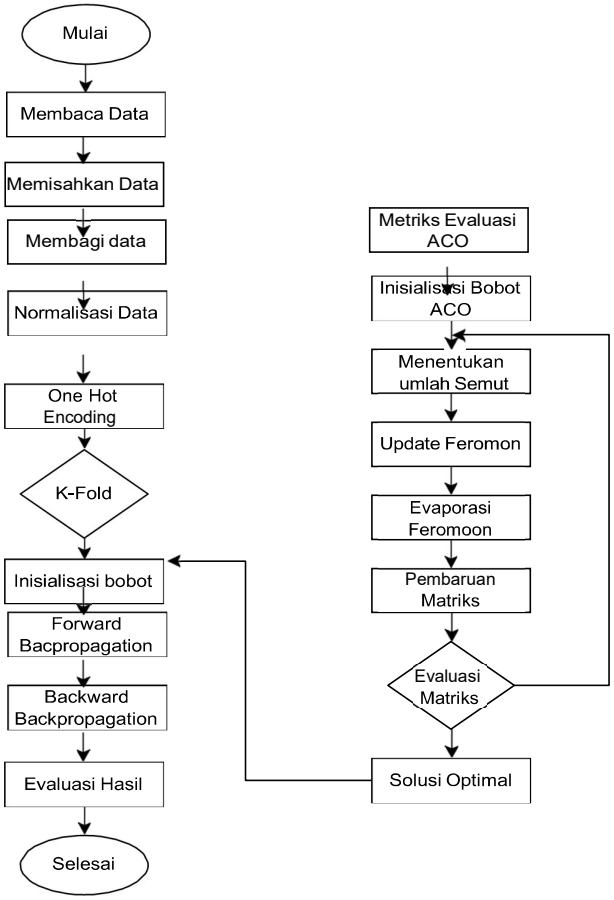
BAB III

HASIL ANALISIS DAN PEMBAHASAN

3.1 Flowchart Metode

Flowchart Metode adalah serangkaian langkah-langkah sistematis yang dilakukan untuk klasifikasi data menggunakan metode *Backpropagation* dan optimasi *Ant Colony Optimization*. *Flowchart Metode* dapat dilihat pada Gambar 3.1



Gambar 3.1 Flowchart Metode

Keterangan flowchart sistem diatas adalah sebagai berikut:

1. Mulai, membaca untuk menampilkan data dan menyimpan pada *DataFrame*. *DataFrame* adalah struktur data dua dimensi yang memungkinkan pengguna untuk menyimpan dan memanipulasi data secara efisien dalam bentuk tabel.
2. Memisahkan data x dan data y yang dimana data x adalah atribut dan data y adalah target.
3. *Split data* adalah proses membagi dataset menjadi dua atau lebih bagian yang terpisah, seperti data latih (*training data*) dan data uji (*test data*).
4. Normalisasi data adalah untuk mengubah skala semua fitur yang ada dalam dataset menjadi skala yang serupa.

5. *One hot encoding* adalah teknik yang digunakan untuk mengubah variabel kategorikal menjadi bentuk biner, sehingga memungkinkan penggunaannya dengan lebih efektif dalam algoritme klasifikasi.
6. *K-Fold Cross-Validation* adalah teknik dalam validasi model yang membagi data menjadi k subset. Dilakukan iterasi sebanyak k kali sehingga setiap subset digunakan untuk pengujian.
7. *Forward propagation* adalah langkah pertama dalam proses pembelajaran (*training*) Jaringan Saraf Tiruan. Tujuannya adalah untuk menghitung nilai *output* dari jaringan berdasarkan input yang diberikan, sehingga dapat memperbaiki nilai-nilai bobot (*weights*) selama proses *training*.
8. *Backward propagation* (penyebaran balik) adalah proses utama dalam pembelajaran Jaringan Saraf Tiruan, khususnya dalam algoritme pembelajaran yang disebut *Backpropagation*.
9. Inisialisasi bobot dan bias adalah langkah kritis dalam pembuatan dan pelatihan jaringan saraf. Bobot (*weights*) dan bias adalah parameter-parameter yang memungkinkan jaringan saraf untuk belajar dari data dan membuat prediksi yang akurat.
10. Menghitung matriks evaluasi adalah proses untuk mengevaluasi kinerja suatu model prediktif berdasarkan perbandingan hasil prediksi dengan nilai yang sebenarnya.
11. Inisialisasi bobot dalam *Algoritme Colony Optimization* (ACO) merujuk pada pengaturan nilai awal dari parameter-parameter yang digunakan seperti jumlah semut, jumlah iterasi, jumlah *node*, laju evaporasi feromon, dan laju deposit feromon.
12. Menentukan jumlah semut dalam algoritme *Ant Colony Optimization* (ACO) merupakan langkah kritis dalam merancang sistem optimasi menggunakan pendekatan ini. Jumlah semut yang tepat dapat berdampak signifikan terhadap kinerja dan efisiensi pencarian solusi.
13. Mengupdate feromon dalam konteks algoritme *Ant Colony Optimization* (ACO) adalah proses di mana nilai-nilai feromon pada jalur yang dilalui semut diperbarui berdasarkan kualitas solusi yang ditemukan oleh semut.
14. Evaporasi feromon adalah mekanisme dalam algoritme *Ant Colony Optimization* (ACO) yang digunakan untuk mengurangi intensitas feromon pada jalur-jalur yang ada seiring waktu. Proses ini penting untuk mencegah jalur-jalur yang tidak optimal dari terus-menerus dipilih oleh semut, dengan memastikan bahwa feromon tidak menumpuk secara berlebihan pada jalur tertentu.
15. Pembaruan matriks feromon adalah proses dalam algoritme *Ant Colony Optimization* (ACO) di mana nilai feromon pada jalur yang dilalui oleh semut diperbarui berdasarkan kualitas solusi yang ditemukan oleh semut
16. Evaluasi model *Ant Colony Optimization* (ACO) adalah proses menilai kinerja algoritme ACO dalam menemukan solusi optimal untuk masalah yang sedang dipecahkan.
17. Mendapatkan hasil akurasi presisi dan *recall* dari algoritme backpropagation dan optimasi *Ant Colony Optimization* (ACO).

3.2 Implementasi Program

Implementasi program adalah proses sebuah program menjadi kode yang bisa digunakan untuk klasifikasi penyakit tanaman padi. Proses ini melibatkan membaca data dari file csv, membagi data, normalisasi data, inisialisasi bobot *Backpropagation*, *k-fold cross validation*, serta optimasi bobot menggunakan *Ant Colony Optimization*.

3.2.1 Pembagian data latih dan data uji

```
Jumlah data latih: 875
Jumlah data uji: 219
```

Gambar 3.2 Pembagian Data

Pada Gambar 3.2 data dibagi menggunakan mode split data dengan 20% data uji dan 80% data latih. dari total 1094 data, untuk data latih sebanyak 875 data dan untuk data uji sebanyak 219 data.

3.2.2 Normalisasi data

```
Data setelah normalisasi:
[[-1.38060615]
 [-1.38060615]
 [-1.38060615]
 ...
 [ 1.67190624]
 [ 1.67190624]
 [ 1.67190624]]
```

Gambar 3.3 Normalisasi Data

Pada Gambar 3.3 digunakan untuk normalisasi data dengan menggunakan objek *standardScaler* dari library *scikit-learn*. Normalisasi data adalah proses mengubah skala fitur dalam dataset sehingga memiliki nilai 0 dan 1. Ini membantu dalam mempercepat konvergensi algoritme pembelajaran mesin dan meningkatkan akurasi model.

3.2.3 K-fold crosss validation

K dibagi menjadi dua subset yang digunakan secara bergantian sebagai data latih dan data uji. dalam penelitian ini, k yang digunakan sebanyak 2 fold.

```
Inisialisasi k-fold cross-validation
Train Index K-fold:
[   1    4    7    9   10   15   18   20   21   22   24   25   28   29
   30   32   36   37   38   39   42   43   44   48   51   52   53   55
   63   64   67   70   71   77   79   83   86   87   93   95   96   97
   98  100  102  105  109  112  113  115  116  122  123  124  127  128
  129  130  131  136  137  138  140  141  143  144  145  149  150  152
  153  155  157  158  162  163  164  166  167  168  169  170  171  173
  174  176  179  182  183  184  188  193  196  197  198  199  203  206
  208  209  210  211  212  213  217  219  221  222  225  227  229  230
  231  232  234  235  237  240  243  249  251  252  253  256  260  261
  263  266  269  270  271  272  275  278  279  284  285  287  288  290
  292  295  296  300  302  303  305  308  312  313  315  316  317  319
  322  324  325  326  327  328  330  332  333  334  337  338  342  343
  346  351  352  353  356  357  359  360  361  362  365  369  373  376
  377  378  379  380  384  391  392  394  396  397  398  399  400  401
  402  405  406  407  409  410  413  416  417  420  421  423  424  425
  427  431  438  440  441  442  444  445  448  450  451  454  456  458
  460  461  462  465  467  468  469  470  475  476  477  485  488  489
  490  492  497  498  499  500  501  503  504  505  506  508  510  511
  513  514  515  517  518  519  520  522  524  526  532  533  536  539
  540  542  543  544  545  547  549  551  556  557  558  562  563  564
```

Gambar 3.4 Train K-fold

Gambar 3.4 menunjukkan indeks baris data yang digunakan untuk melatih model pada setiap fold. Seperti pada gambar 6 data latih mencakup indeks 1, 4, 7, 9 dan seterusnya hingga indeks yang ditentukan.

```
Test Index k-fold:
[  0   2   3   5   6   8  11  12  13  14  16  17  19  23
  26  27  31  33  34  35  40  41  45  46  47  49  50  54
  56  57  58  59  60  61  62  65  66  68  69  72  73  74
  75  76  78  80  81  82  84  85  88  89  90  91  92  94
  99 101 103 104 106 107 108 110 111 114 117 118 119 120
 121 125 126 132 133 134 135 139 142 146 147 148 151 154
 156 159 160 161 165 172 175 177 178 180 181 185 186 187
 189 190 191 192 194 195 200 201 202 204 205 207 214 215
 216 218 220 223 224 226 228 233 236 238 239 241 242 244
 245 246 247 248 250 254 255 257 258 259 262 264 265 267
 268 273 274 276 277 280 281 282 283 286 289 291 293 294
 297 298 299 301 304 306 307 309 310 311 314 318 320 321
 323 329 331 335 336 339 340 341 344 345 347 348 349 350
 354 355 358 363 364 366 367 368 370 371 372 374 375 381
 382 383 385 386 387 388 389 390 393 395 403 404 408 411
 412 414 415 418 419 422 426 428 429 430 432 433 434 435
 436 437 439 443 446 447 449 452 453 455 457 459 463 464
 466 471 472 473 474 478 479 480 481 482 483 484 486 487
 491 493 494 495 496 502 507 509 512 516 521 523 525 527
 528 529 530 531 534 535 537 538 541 546 548 550 552 553
 554 555 559 560 561 567 571 574 576 577 578 579 581 582
 584 587 589 593 594 597 599 600 602 604 607 612 613 616
 619 623 625 629 631 632 635 638 639 640 641 644 650 651
 653 655 658 659 661 664 667 671 672 673 674 676 677 678
```

Gambar 3.5 Test K-fold

Pada Gambar 3.5 menunjukkan bahwa *indeks test* merupakan data yang tidak termasuk dalam *index train* yang disebut dengan data uji. Data uji merupakan elemen penting dalam evaluasi model *Machine Learning*, memberikan gambaran yang lebih objektif tentang performanya dan kemampuannya untuk menggeneralisasi ke data baru. Dengan menggunakan data uji, kita dapat memastikan model tidak hanya berkinerja baik pada data latihan, tetapi juga pada data baru yang belum pernah dilihat sebelumnya.

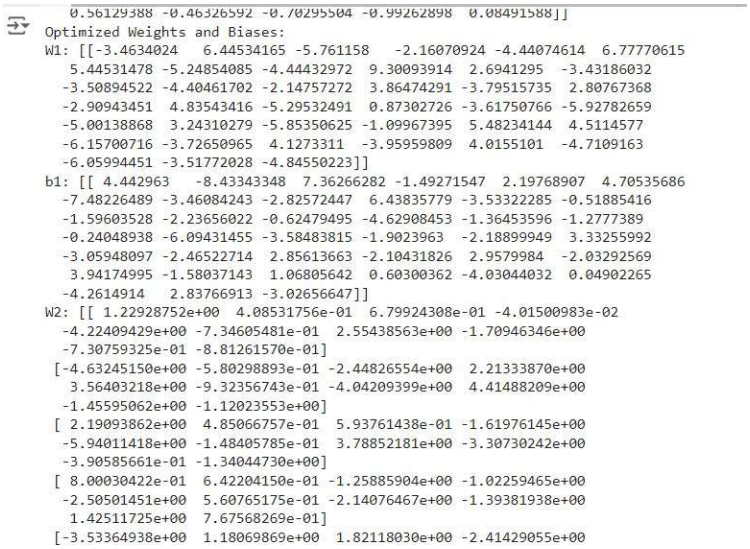
3.2.4 Inisialisasi bobot dan bias

```
Menginisialisasi nilai bobot dan bias dari Neural Network
W1_initial: [[ 0.40996297 -0.11031897  0.07007342  0.21783153  0.69824189  0.65111738
  -0.85741455 -0.02207042  0.15916658  0.40255433  0.64623419  0.83537571
  -0.62685876 -0.80918767  0.74434031]]
b1_initial: [[-0.66514747  0.42211954 -0.82934131 -0.67331829 -0.62668865 -0.62724821
   0.88534382 -0.85601715 -0.57016283  0.91975921 -0.552272   -0.16843468
  -0.29557185 -0.33283625  0.5399629 ]]
W2_initial: [[-0.70591757  0.5762349   0.99567096  0.50292829  0.22814992 -0.53811024
  -0.78865205  0.40603609 -0.27985372  0.70511119  0.9468194   0.3141115
  -0.67302902  0.92593024  0.28292708 -0.55836301 -0.01600683  0.69695628
   0.39384653  0.91482551  0.26005018 -0.27206406 -0.82040285  0.62913827
  -0.79015108]
 [-0.64147799 -0.19976269 -0.57970718  0.41391886  0.23392304 -0.20857582
  -0.56042659 -0.74769681 -0.55665659  0.23668659  0.39739579 -0.40704473
   0.56663576  0.69974905  0.92919333  0.12747782 -0.82593992  0.17874755
  -0.15705408 -0.52169349 -0.11416799 -0.06213294 -0.06373973 -0.51831171
  -0.16938652]
```

Gambar 3.6 Inisialisasi Bobot

Pada Gambar 3.6 menunjukkan bahwa matriks w1_initial, b1_initial, w2_initial, b2_initial, w3_initial, dan b3_initial merupakan elemen penting dalam Jaringan Saraf Tiruan, mewakili nilai awal bobot dan bias pada setiap lapisan. Inisialisasi yang tepat dapat mempercepat proses pembelajaran, meningkatkan performa akhir model.

3.2.5 Optimasi bobot dan bias



```
  0.56129388 -0.46326592 -0.70295504 -0.99262898  0.08491588]]
Optimized Weights and Biases:
W1: [[-3.4634024   6.44534165 -5.761158   -2.16070924 -4.44074614  6.77770615
   5.44531478 -5.24854085 -4.44432972  9.30093914  2.6941295  -3.43186032
  -3.50894522 -4.40461702 -2.14757272  3.86474291 -3.79515735  2.80767368
  -2.90943451  4.83543416 -5.29532491  0.87302726 -3.61750766 -5.92782659
  -5.00138868  3.24310279 -5.85350625 -1.09967395  5.48234144  4.5114577
  -6.15700716 -3.72650965  4.1273311  -3.95959809  4.0155101  -4.7109163
  -6.05994451 -3.51772028 -4.84550223]]
b1: [[ 4.442963   -8.43343348  7.36266282 -1.49271547  2.19768907  4.70535686
  -7.48226489 -3.46084243 -2.82572447  6.43835779 -3.53322285 -0.51885416
  -1.59603528 -2.23656022 -0.62479495 -4.62908453 -1.36453596 -1.2777389
  -0.24048938 -6.09431455 -3.58483815 -1.9023963  -2.18899949  3.33255992
  -3.05948097 -2.46522714  2.85613663 -2.10431826  2.9579984  -2.03292569
   3.94174995 -1.58037143  1.06805642  0.60300362 -4.03044032  0.04902265
  -4.2614914   2.83766913 -3.02656647]]
W2: [[ 1.22928752e+00  4.08531756e-01  6.79924308e-01 -4.01500983e-02
  -4.22409429e+00 -7.34605481e-01  2.55438563e+00 -1.70946346e+00
  -7.30759325e-01 -8.81261570e-01]
 [-4.63245150e+00 -5.80298893e-01 -2.44826554e+00  2.21333870e+00
   3.56403218e+00 -9.32356743e-01 -4.04209399e+00  4.41488209e+00
  -1.45595062e+00 -1.12023553e+00]
 [ 2.19093862e+00  4.85066757e-01  5.93761438e-01 -1.61976145e+00
  -5.94011418e+00 -1.48405785e-01  3.78852181e+00 -3.30730242e+00
  -3.90585661e-01 -1.34044730e+00]
 [ 8.00030422e-01  6.42204150e-01 -1.25885904e+00 -1.02259465e+00
  -2.50501451e+00  5.60765175e-01 -2.14076467e+00 -1.39381938e+00
   1.42511725e+00  7.67568269e-01]
 [-3.53364938e+00  1.18069869e+00  1.82118030e+00 -2.41429055e+00
```

Gambar 3.7 Optimasi Bobot

Pada Gambar 3.7 menunjukkan bahwa bobot dan bias merupakan parameter penting dalam Jaringan Saraf Tiruan yang dioptimalkan selama proses pelatihan. Optimasi ini bertujuan untuk meminimalkan kesalahan klasifikasi dan meningkatkan akurasi. Nilai-nilai bobot dan bias mencerminkan penyesuaian parameter model terhadap data latihan, memungkinkannya untuk mempelajari pola dan hubungan yang kompleks dalam data dan menghasilkan klasifikasi yang lebih akurat.

3.2.6 Memperbarui dan menghitung kesalahan nilai bobot dan bias

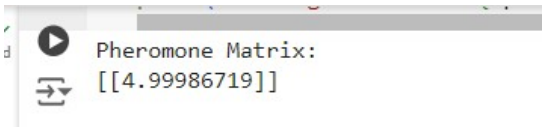
```
Melatih BPNN, memperbarui nilai bobot dan bias, menghitung kesalahan
Epoch 0, Loss: 0.0007930529363389238
Epoch 100, Loss: 0.0007699390065351823
Epoch 200, Loss: 0.0007567078139057024
Epoch 300, Loss: 0.000745336975008863
Epoch 400, Loss: 0.0007347649461538723
```

Gambar 3.8 Memperbarui dan Menghitung Kesalahan

Pada Gambar 3.8 merupakan hasil pelatihan sebanyak 500 perulangan *(epoch)* menunjukkan perubahan nilai bobot dan bias yang terus menerus untuk meminimalkan nilai eror *(loss)*. Pada pelatihan awal, nilai *loss* sebesar 0,00079 hingga pelatihan akhir sebesar 0,00073 pada perulangan ke 400. Dengan demikian, model dapat meningkatkan akurasi prediksi dengan mengurangi nilai *loss* yang terus-menerus berkurang selama pelatihan.

3.2.7 Matriks pheromone

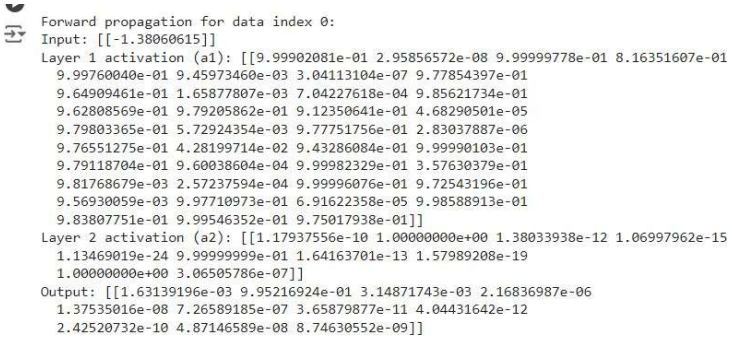
Matriks feromon *(pheromone matrix)* adalah konsep yang berasal dari algoritme koloni semut *Ant Colony Optimization* (ACO). Dalam ACO, matriks feromon digunakan untuk mempresentasikan tingkat feromon yang terdapat pada berbagai jalur yang bisa dilalui oleh semut-semut buatan dalam algoritme tersebut. Feromon adalah zat kimia yang dikeluarkan oleh semut-semut nyata untuk berkomunikasi dan menandai jalur yang telah mereka tempuh.



Gambar 3.9 Matriks Peromon

Pada Gambar 3.9 merupakan matriks feromon adalah elemen kunci dalam algoritme optimasi koloni semut (ACO). Matriks ini digunakan untuk menyimpan tingkat feromon yang menandakan kualitas jalur yang telah dieksplorasi oleh semut-semut buatan. Pada gambar menghasilkan jumlah matriks peromon sebanyak 4,999.

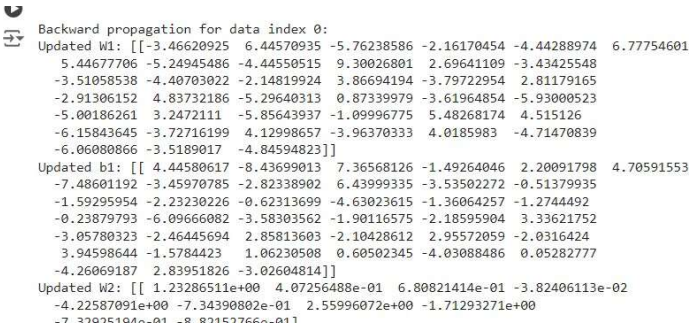
3.2.8 Forward backropagation



Gambar 3.10 Forward

Gambar 3.10 menunjukkan hasil forward dalam metode backpropagation. Proses dalam forward adalah dengan input data melewati hidden layer 1 dan hidden layer 2 untuk menghasilkan output.

3.2.9 Backward backpropagation

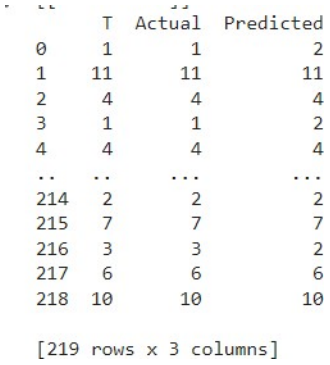


Gambar 3.11 Backward

Gambar 3.11 menunjukkan hasil dari proses backward propagation pada jaringan syaraf tiruan. Proses ini dilakukan untuk memperbarui bobot dan bias jaringan berdasarkan error yang dihitung dari hasil forward propagation. Bobot $W1,W2,W3W1,W2,W3$ dan bias $b1,b2,b3b1,b2,b3$ diperbarui dengan nilai baru untuk mengurangi error dan meningkatkan akurasi model. Nilai-nilai yang ditampilkan adalah bobot dan bias yang telah disesuaikan melalui algoritme *Backpropagation*, di mana $W1,W2,W3W1,W2,W3$ mewakili matriks bobot pada masing-masing lapisan, sementara $b1,b2,b3b1,b2,b3$ mewakili vektor bias. Pembaruan ini memastikan bahwa

Jaringan Saraf Tiruan dapat belajar dari data dan memperbaiki klasifikasi berdasarkan input yang diberikan.

3.2.10 Menampilkan Data Uji dan Data Latih



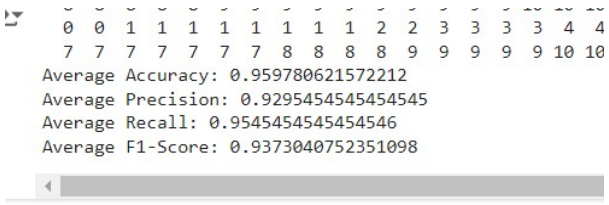
	T	Actual	Predicted
0	1	1	2
1	11	11	11
2	4	4	4
3	1	1	2
4	4	4	4
..	..	...	...
214	2	2	2
215	7	7	7
216	3	3	2
217	6	6	6
218	10	10	10

[219 rows x 3 columns]

Gambar 3. 12 Frame untuk Data Uji dan Data Latih

Pada gambar 3.12 menunjukkan hasil data uji dan data prediksi. Kolom "T" menampilkan hasil kelas/target, kolom "*Actual*" menampilkan hasil untuk data asli, sedangkan "*Predicted*" menampilkan hasil untuk data yang telah di klasifikasi.

3.2.11 Hasil Evaluasi



```
  0  0  1  1  1  1  1  1  1  1  2  2  3  3  3  3  4  4
  7  7  7  7  7  7  7  8  8  8  8  9  9  9  9  9 10 10
Average Accuracy: 0.959780621572212
Average Precision: 0.9295454545454545
Average Recall: 0.9545454545454546
Average F1-Score: 0.9373040752351098
```

Gambar 3.13 Hasil Evaluasi

Pada Gambar 3.13 menunjukkan hasil evaluasi model pada akurasi, presisi, *recall*, dan *f1-score*. Akurasi menghasilkan sebesar 95%, presisi menghasilkan sebesar 92%, untuk recall menghasilkan sebesar 95%, dan f1-score menghasilkan sebesar 93%.

3.3 Pengujian

Setiap jenis pengujian akan dilakukan dengan menggunakan nilai parameter awal yang sama. Hal ini dilakukan untuk memastikan konsistensi dalam evaluasi dan memastikan bahwa semua aspek diuji berdasarkan kondisi awal yang serupa.

3.3.1 Parameter Awal *Backpropagation*

Parameter awal merujuk pada nilai awal yang ditetapkan untuk variabel atau parameter tertentu sebelum proses pengujian.

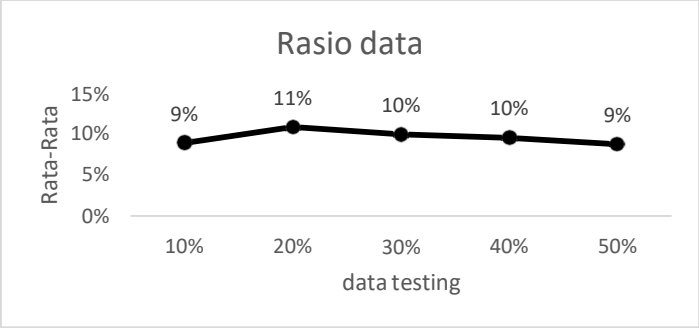
Tabel 3.1 Parameter Awal

rasio	k-fold	LR	epoch	hidden 1	hidden 2	akurasi
20%	4	0,07	500	55	65	8%

Tabel 3.1 menunjukan parameter awal yang akan digunakan pada pengujian, yaitu rasio data 20%, k-fold berjumlah 4, *Learning Rate* bernilai 0,07, epoch bernilai 500, hidden 1 bernilai 55, dan hidden 2 bernilai 65 dengan akurasi 8%.

3.3.2 Rasio Data

Rasio data digunakan untuk membagi data menjadi data latih (*training data)* dan data uji *(test data)*.

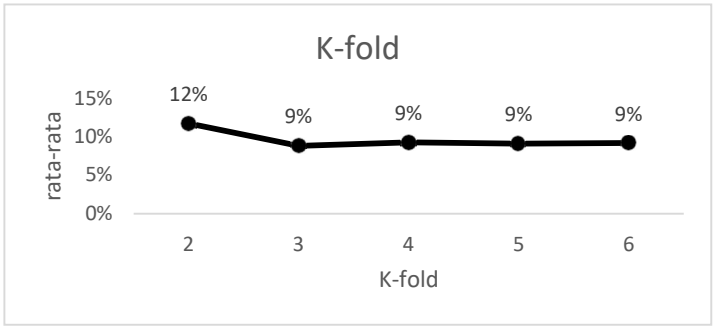


Gambar 3.14 *Split Data*

Pada Gambar 3.14 menunjukkan bahwa akurasi tertinggi untuk pembagian data adalah sebsar 80% data latih dan 20% data uji yang menghasilkan akurasi sebesar 11%.

3.3.3 K-fold

K-fold cross validation merupakan teknik membagi dataset menjadi K bagian *(fold)* yang kira-kira memiliki ukuran yang sama. Proses ini kemudian melibatkan pelatihan dan pengujian model K kali, di mana pada setiap iterasi, satu fold digunakan sebagai data uji dan K-*fold* lainnya digunakan sebagai data latih.

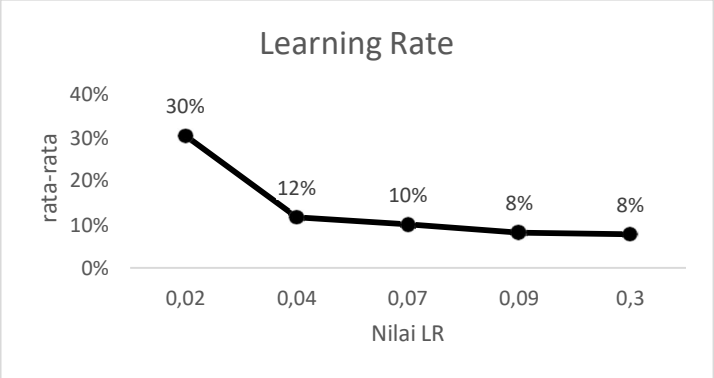


Gambar 3.15 *K-fold*

Pada Gambar 3.15 menunjukkan bahwa semakin tinggi nilai k-fold nya, maka semakin rendah akurasi yang didapatkan. Dalam penelitian ini nilai k-fold terbaik menghasilkan akurasi sebesar 12% dengan *k-fold* bernilai 2.

3.3.4 Learning Rate

Learning rate menentukan ukuran langkah yang diambil untuk memperbarui bobot model pada setiap iterasi pelatihan. Nilai *learning rate* yang tepat sangat penting untuk memastikan bahwa model dapat belajar secara efektif dan efisien.

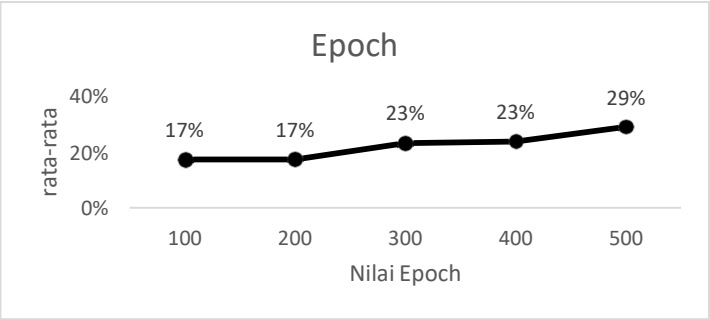


Gambar 3.16 *Learning Rate*

Pada Gambar 3.16 menunjukkan bahwa semakin rendah nilai nya, maka semakin tinggi akurasi yang didapatkan. Dalam penelitian ini nilai *learning rate* terbaik menghasilkan akurasi sebesar 30% dengan *learning rate* bernilai 0,02.

3.3.5 Epoch

Epoch merupakan proses *training* dari *Neural Network* sampai kembali pada tahap awal dengan satu putaran ketika seluruh dtaset melalui proses tersebut cukup dengan satu *epoch* maka terlalu besar dan prosesnya akan berat pada dataset karena data yang dipakai terlalu banyak.

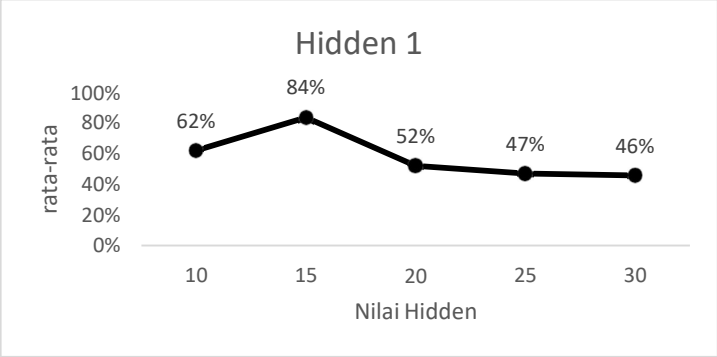


Gambar 3.17 Epoch

Pada Gambar 3.17 menunjukkan bahwa semakin tinggi nilai *epoch* nya, maka semakin tinggi akurasi yang didapatkan. Dalam penelitian ini nilai *epoch* terbaik menghasilkan akurasi sebesar 29% dengan *epoch* bernilai 500.

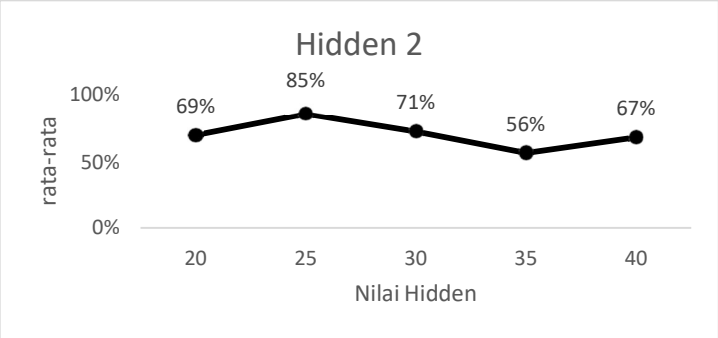
3.3.6 *Hidden* Layer

Hidden layer (lapisan tersembunyi) adalah salah satu komponen utama dalam Jaringan Saraf Tiruan (*Neural Network*). Lapisan ini terletak di antara *input layer* (lapisan masukan) dan *output layer* (lapisan keluaran).



Gambar 3.18 Hidden 1

Menambah jumlah *hidden layer* bisa meningkatkan akurasi model, tetapi tidak selalu. Peningkatan ini berlaku hanya sampai batas tertentu, Pada pengujian ini jumlah *hidden* 1 terbaik bernilai 15 dengan akurasi sebesar 84%.



Gambar 3.19 Hidden 2

Pada Gambar 3.19 menunjukkan bahwa nilai *hidden layer* 2 yang terbaik bernilai 25 dengan akurasi sebesar 85%.

3.3.7 Parameter Akhir *Backpropagation*

Parameter akhir merupakan nilai yang telah melewati proses pengujian sehingga mendapatkan akurasi yang lebih baik.

Tabel 3.2 Parameter Akhir

rasio	k-fold	LR	epoch	hidden 1	hidden 2	akurasi
20%	2	0,02	500	15	25	0,80073

Tabel 3.2 menunjukan parameter akhir yang akan digunakan pada pengujian, yaitu rasio data 20%, *k-fold* berjumlah 2, *Learning Rate* bernilai 0,02, *epoch* bernilai 500, *hidden* 1 bernilai 15, dan *hidden* 2 bernilai 25 dengan akurasi 80%.

3.3.8 Parameter Awal *Ant Colony Optimization*

Parameter awal *Ant Colony Optimization* merupakan nilai yang telah ditentukan untuk optimasi algoritme *Ant Colony Optimization*.

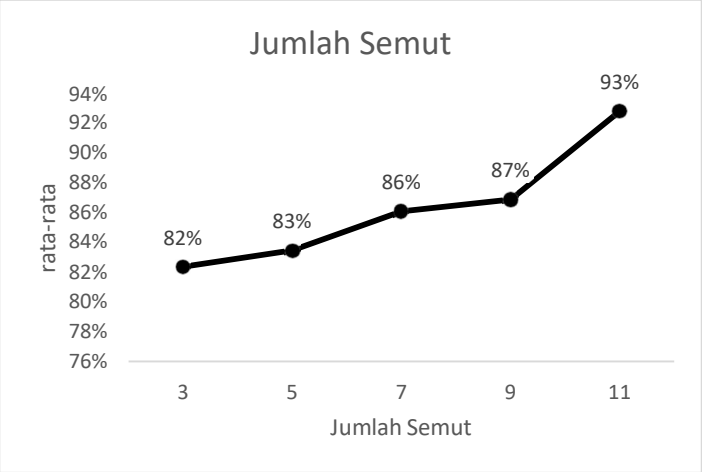
Tabel 3.3 Parameter Awal Optimasi

semut	iterasi	akurasi
10	100	91%

Tabel 3.3 menunjukan parameter awal yang akan digunakan pada pengujian optimasi, yaitu jumlah semut bernilai 10, iterasi bernilai 100 dengan akurasi 91%.

3.3.9 Jumlah Semut

Jumlah semut dalam *Ant Colony Optimization* (ACO) merujuk pada jumlah agen (semut buatan) yang digunakan dalam setiap iterasi algoritme untuk mencari solusi. Setiap semut secara independen menjelajahi ruang solusi dengan mengikuti jejak feromon dan heuristik yang ada, serta membangun solusi potensial untuk masalah yang sedang dihadapi.

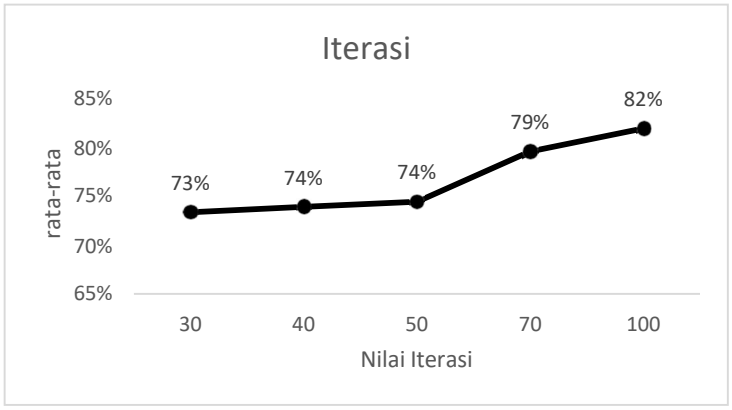


Gambar 3.20 Jumlah Semut

Gambar 3.20 berguna untuk semut dalam memperluas ruang pencarian untuk menemukan jalur yang lebih baik pada setiap iterasi berikutnya. Pada penelitian ini, jumlah semut 11 lebih baik dalam membantu semut untuk menemukan kemungkinan jalur lebih baik lainnya dengan akurasi sebesar 93%.

3.3.10 Iterasi

Iterasi dalam *Ant Colony Optimization* (ACO) adalah satu siklus penuh dari algoritme di mana sejumlah semut (agen) membangun solusi, jejak feromon diperbarui, dan penilaian terhadap solusi dilakukan.



Gambar 3.21 *Iterasi*

Gambar 3.21 menunjukkan bahwa semakin besar jumlah iterasi yang dilakukan maka solusinya semakin baik. Pada penelitian ini iterasi terbaik bernilai 100 dengan akurasi sebesar 82%.

3.3.11 Parameter Akhir *Ant Colony Optimization*

Parameter akhir *Ant Colony Optimization* merupakan nilai yang telah melewati proses pengujian sehingga mendapatkan akurasi yang lebih baik.

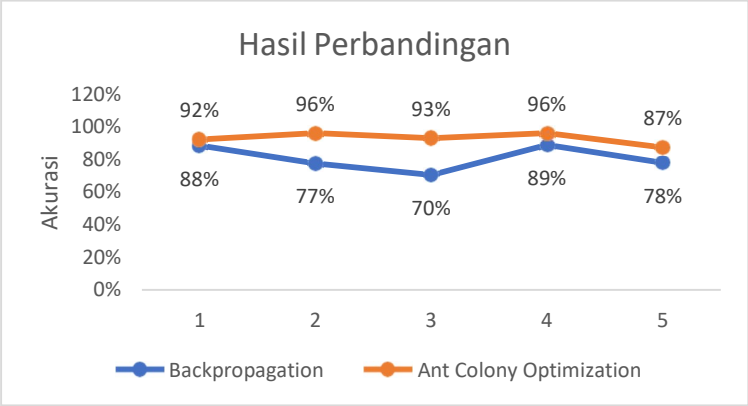
Tabel 3.4 Parameter Akhir Optimasi

semut	iterasi	akurasi
11	100	93%

Tabel 3.4 menunjukan parameter akhir yang telah melewati pengujian optimasi dengan nilai terbaik, yaitu jumlah semut bernilai 11, iterasi bernilai 100 dengan akurasi 93%.

3.3.12 Perbandingan Hasil

Perbandingan hasil adalah analisis untuk melihat perbedaan atau persamaan antara dua metode yang digunakan yaitu metode *Backpropagation* dan metode *Backpropagation* dengan optimasi *Ant Colony Optimization*.



Gambar 3.22 Hasil Perbandingan

Pada Gambar 3.22 Algoritme *Backpropagation* mengalami peningkatan akurasi setelah dioptimasi menggunakan metode *Ant Colony Optimization*. Optimalisasi ini memungkinkan pencarian bobot dan bias yang lebih efektif, sehingga model *Neural Network* dapat menghasilkan prediksi yang lebih tepat dan akurat.

3.4 Hasil

Hasil pengujian pada kedua metode di ujikan sebanyak 5 kali, lalu dihitung rata-ratanya. bahwa *Backpropagation*-ACO memiliki akurasi lebih tinggi dibandingkan *Backpropagation* dengan akurasi sebesar 80%. Sedangkan, *Backpropagation*-ACO mendapatkan akurasi sebesar 92%. Sehingga dapat dikatakan ACO berhasil menaikkan akurasi *Backpropagation* sebesar 12%.