

<?php

```
/**
 * 231232028 - Falmesino Abdul Hamid
 * Logistic Regresssion dengan Gradient Descent Regularized (L2)
 *
 * Dataset diasumsikan berupa array asosiatif:
 * - X: matriks fitur (m x n)
 * - y: label (0 atau 1)
 */

function sigmoid($z) {
    return 1 / (1 + exp(-$z));
}

function logisticRegressionGD($X, $y, $learning_rate = 0.01, $iterations = 1000, $lambda = 0.1) {
    $m = count($y); // Jumlah data
    $n = count($X[0]); // Jumlah fitur
    $theta = array_fill(0, $n, 0.0); // Inisialisasi parameter model

    // Gradient descent
    for ($iter = 0; $iter < $iterations; $iter++) {
        $gradients = array_fill(0, $n, 0.0);

        // Hitung gradient untuk setiap data
        for ($i = 0; $i < $m; $i++) {
            $z = 0;
            for ($j = 0; $j < $n; $j++) {
                $z += $theta[$j] * $X[$i][$j];
            }
            $h = sigmoid($z);
            $error = $h - $y[$i];
            for ($j = 0; $j < $n; $j++) {
                $gradients[$j] += $error * $X[$i][$j];
            }
        }

        // Update parameter dengan regulisasi L2 (kecuali theta[0] bias)
        for ($j = 0; $j < $n; $j++) {
            $reg = ($j == 0) ? 0 : ($lambda / $m) * $theta[$j];
            $theta[$j] -= $learning_rate * (($gradients[$j] / $m) + $reg);
        }
    }

    return $theta;
}

// Contoh data (m = 4, n = 3)
$X = [
    [1, 2.5, 3.1], // Baris pertama data (termasuk fitur bias jika diperlukan)
    [1, 3.0, 3.8],
    [1, 2.8, 3.0],
    [1, 3.2, 3.9]
];
$y = [0, 1, 0, 1];
$theta = logisticRegressionGD($X, $y, 0.05, 20000, 0.05);

echo "Parameter model (theta):\n";
print_r($theta);

/**
 * Analisis
 * - Setiap iterasi memproses m data dengan fitur -> O(m * n) per iterasi.
 * - Total iterasi T menghasilkan kompleksitas waktu O(T * m * n).
 * - Ruang O(n) untuk theta dan gradien.
 */
```

?>



EXPLOLER

SDA-PRAKTIKUM-14

- 1.php
- 2.php
- 3.php
- 4.php
- 5.php

OUTLINE

1.php

```
1 <?php
2
3 /**
4  * 231232028 - Falmesino Abdul Hamid
5  * Logistic Regresssion dengan Gradient Descent Regularized (L2)
6  *
7  * Dataset diasumsikan berupa array asosiatif:
8  * - X: matriks fitur (m x n)
9  * - y: label (0 atau 1)
10 *
11
12 function sigmoid($z) {
13     return 1 / (1 + exp(-$z));
14 }
15
16 function logisticRegressionGD($X, $y, $learning_rate = 0.01, $iterations = 1000, $lambda = 0.1) {
17     $m = count($y); // Jumlah data
18     $n = count($X[0]); // Jumlah fitur
19     $theta = array_fill(0, $n, 0.0); // Inisialisasi parameter model
20
21     // Gradient descent
22     for ($iter = 0; $iter < $iterations; $iter++) {
23         $gradients = array_fill(0, $n, 0.0);
24
25         // Hitung gradient untuk setiap data
26         for ($i = 0; $i < $m; $i++) {
27             $z = 0;
28             for ($j = 0; $j < $n; $j++) {
29                 $z += $theta[$j] * $X[$i][$j];
30             }
31             $h = sigmoid($z);
32             $error = $h - $y[$i];
33             for ($j = 0; $j < $n; $j++) {
34                 $gradients[$j] += $error * $X[$i][$j];
35             }
36         }
37
38         // Update parameter dengan regulisasi L2 (kecuali theta[0] bias)
39         for ($j = 0; $j < $n; $j++) {
40             $reg = ($j == 0) ? 0 : ($lambda / $m) * $theta[$j];
41             $theta[$j] -= $learning_rate * (($gradients[$j] / $m) + $reg);
42         }
43     }
44
45     return $theta;
46 }
47
48 // Contoh data (m = 4, n = 3)
49 $X = [
50     [1, 2.5, 3.1], // Baris pertama data (termasuk fitur bias jika diperlukan)
51     [1, 2.0, 2.0]
```

PORTS

GITLENS

TERMINAL

zsh

```
➔ sda-praktikum-14 git:(main) ls -la
total 40
drwxr-xr-x  8 falmesino  staff   256 Feb  8 02:36 .
drwxr-xr-x 38 falmesino  staff  1216 Feb  7 22:45 ..
drwxr-xr-x 12 falmesino  staff   384 Feb  8 03:07 .git
-rw-r--r--  1 falmesino  staff  1832 Feb  8 00:38 1.php
-rw-r--r--  1 falmesino  staff  3728 Feb  8 02:13 2.php
-rw-r--r--  1 falmesino  staff  1557 Feb  8 02:20 3.php
-rw-r--r--  1 falmesino  staff  2728 Feb  8 03:07 4.php
-rw-r--r--  1 falmesino  staff  1811 Feb  8 02:45 5.php

➔ sda-praktikum-14 git:(main) valet php 1.php
Parameter model (theta):
Array
(
    [0] => -16.062525491614
    [1] => 1.4083091811625
    [2] => 3.497920854293
)
```

Falmesino Abdul Hamid (34 minutes ago)

Ln 20, Col 1

Spaces: 2

UTF-8

LF

{ } PHP

Colorize

Colorize: 0 variables

Prettier



```
<?php

/**
 * 231232028 - Falmesino Abdul Hamid
 * Contoh Implementasi A* Sederhana
 */

class Node {
    public $x;
    public $y;
    public $g; // cost from start
    public $h; // heuristic cost to goal
    public $f; // total cost = g + h
    public $parent;

    public function __construct($x, $y, $g = 0, $h = 0, $parent = null) {
        $this->x = $x;
        $this->y = $y;
        $this->g = $g;
        $this->h = $h;
        $this->f = $g + $h;
        $this->parent = $parent;
    }
}

// Fungsi heuristic (menggunakan jarak Manhattan)
function heuristic($node, $goal) {
    return abs($node->x - $goal->x) + abs($node->y - $goal->y);
}

// Fungsi untuk mendapatkan tetangga (4 arah)
function getNeighbors($node, $grid) {
    $neighbors = [];
    $directions = [[0,1], [1,0], [0,-1], [-1,0]];
    foreach ($directions as $d) {
        $nx = $node->x + $d[0];
        $ny = $node->y + $d[1];

        if (isset($grid[$ny][$nx]) && $grid[$ny][$nx] == 0) { // 0 = jalur bebas
            $neighbors[] = new Node($nx, $ny);
        }
    }
    return $neighbors;
}

// Implementasi A*
function aStar($grid, $start, $goal) {
    $openSet = [];
    $closedSet = [];
    $startNode = new Node($start[0], $start[1]);
    $startNode->h = heuristic($startNode, new Node($goal[0], $goal[1]));
    $startNode->f = $startNode->g + $startNode->h;
    $openSet[] = $startNode;

    while(!empty($openSet)) {
        // Pilih node dengan nilai f terkecil
        usort($openSet, function($a, $b) {
            return $a->f <=> $b->f;
        });
        $current = array_shift($openSet);

        // Jika mencapai tujuan, rekonstruksi jalur
        if ($current->x == $goal[0] && $current->y == $goal[1]) {
            $path = [];
            while ($current != null) {
                $path[] = [$current->x, $current->y];
                $current = $current->parent;
            }
            return array_reverse($path);
        }

        $closedSet[] = $current;
        foreach(getNeighbors($current, $grid) as $neighbor) {
            // Jika sudah ada di closedSet, lewati
            $skip = false;
            foreach($closedSet as $closed) {
                if ($closed->x == $neighbor->x && $closed->y == $neighbor->y) {
                    $skip = true;
                    break;
                }
            }

            if ($skip) continue;

            $tentative_g = $current->g + 1; // Asumsi jarak antar tetangga = 1

            // Jika neighbor sudah ada di openSet dengan nilai g lebih kecil, lewati
            $inOpen = false;
            foreach ($openSet as $node) {
                if ($node->x == $neighbor->x && $node->y == $neighbor->y) {
                    $inOpen = true;
                    if ($tentative_g >= $node->g) {
                        $skip = true;
                    }
                    break;
                }
            }

            $neighbor->g = $tentative_g;
            $neighbor->h = heuristic($neighbor, new Node($goal[0], $goal[1]));
            $neighbor->f = $neighbor->g + $neighbor->h;
            $neighbor->parent = $current;
            $openSet[] = $neighbor;
        }
    }
    return null; // Tidak ditemukan jalur
}

// Contoh grid (0 = bebas, 1 = halangan)
$grid = [
    [0, 0, 0, 0, 0],
    [0, 1, 1, 1, 0],
    [0, 0, 0, 1, 0],
    [0, 1, 0, 0, 0],
    [0, 0, 0, 1, 0]
];
$start = [0, 0];
$goal = [4, 4];

$path = aStar($grid, $start, $goal);

if ($path) {
    echo "Jalur ditemukan:\n";
    foreach ($path as $p) {
        echo "(" . $p[0] . ", " . $p[1] . ") ";
    }
    echo "\n";
} else {
    echo "Jalur tidak ditemukan.\n";
}

/**
 * Analisis
 * - Kompleksitas waktu bergantung pada jumlah node dan heuristik, secara umum
 *   O(n log n) dengan penggunaan priority queue (di sini disort setiap iterasi).
 * - Kompleksitas ruang O(n) untuk openSet dan closedSet.
 */
?>
```



EXPLOLER

SDA-PRAKTIKUM-14

- 1.php
- 2.php
- 3.php
- 4.php
- 5.php

OUTLINE

2.php

```
1 <?php
45
46 // Implementasi A*
47 function aStar($grid, $start, $goal) {
48     $openSet = [];
49     $closedSet = [];
50     $startNode = new Node($start[0], $start[1]);
51     $startNode->h = heuristic($startNode, new Node($goal[0], $goal[1]));
52     $startNode->f = $startNode->g + $startNode->h;
53     $openSet[] = $startNode;
54
55     while(!empty($openSet)) {
56         // Pilih node dengan nilai f terkecil
57         usort($openSet, function($a, $b) {
58             return $a->f <=> $b->f;
59         });
60         $current = array_shift($openSet);
61
62         // Jika mencapai tujuan, rekonstruksi jalur
63         if ($current->x == $goal[0] && $current->y == $goal[1]) {
64             $path = [];
65             while ($current != null) {
66                 $path[] = [$current->x, $current->y];
67                 $current = $current->parent;
68             }
69             return array_reverse($path);
70         }
71
72         $closedSet[] = $current;
73         foreach(getNeighbors($current, $grid) as $neighbor) {
74             // Jika sudah ada di closedSet, lewati
75             $skip = false;
76             foreach($closedSet as $closed) {
77                 if ($closed->x == $neighbor->x && $closed->y == $neighbor->y) {
78                     $skip = true;
79                     break;
80                 }
81             }
82
83             if ($skip) continue;
84
85             $tentative_g = $current->g + 1; // Asumsi jarak antar tetangga = 1
86
87             // Jika neighbor sudah ada di openSet dengan nilai g lebih kecil, lewati
88             $inOpen = false;
89
90             foreach ($openSet as $node) {
91                 if ($node->x == $neighbor->x && $node->y == $neighbor->y) {
92                     $inOpen = true;
93                     if ($tentative_g >= $node->g) {
94                         $skin = true;
```

PORTS

GITLENS

TERMINAL

zsh

```
➔ sda-praktikum-14 git:(main) ls -la
total 40
drwxr-xr-x  8 falmesino  staff   256 Feb  8 02:36 .
drwxr-xr-x 38 falmesino  staff  1216 Feb  7 22:45 ..
drwxr-xr-x 12 falmesino  staff   384 Feb  8 03:07 .git
-rw-r--r--  1 falmesino  staff  1832 Feb  8 00:38 1.php
-rw-r--r--  1 falmesino  staff  3728 Feb  8 02:13 2.php
-rw-r--r--  1 falmesino  staff  1557 Feb  8 02:20 3.php
-rw-r--r--  1 falmesino  staff  2728 Feb  8 03:07 4.php
-rw-r--r--  1 falmesino  staff  1811 Feb  8 02:45 5.php

➔ sda-praktikum-14 git:(main) valet php 2.php
Jalur ditemukan:
(0, 0) (0, 1) (0, 2) (1, 2) (2, 2) (2, 3) (3, 3) (4, 3) (4, 4)

➔ sda-praktikum-14 git:(main)
```

Falmesino Abdul Hamid (35 minutes ago)

Ln 59, Col 8

Spaces: 2

UTF-8

LF

{ } PHP

Colorize

Colorize: 0 variables

Prettier



<?php

```
/**
 * 231232028 - Falmesino Abdul Hamid
 * Simulasi sederhana MapReduce untuk menghitung total
 * penjualan per kategori.
 */

// Simulasi data transaksi (kategori, nilai penjualan)
$transactions = [
    ['kategori' => 'Elektronik', 'nilai' => 100],
    ['kategori' => 'Pakaian', 'nilai' => 50],
    ['kategori' => 'Elektronik', 'nilai' => 200],
    ['kategori' => 'Makanan', 'nilai' => 30],
    ['kategori' => 'Pakaian', 'nilai' => 70],
    ['kategori' => 'Makanan', 'nilai' => 45],
];

// Fungsi Mapper: mengelompokkan data per kategori
function mapper($transactions) {
    $mapped = [];
    foreach ($transactions as $trans) {
        $key = $trans['kategori'];
        $value = $trans['nilai'];
        if (!isset($mapped[$key])) {
            $mapped[$key] = [];
        }
        $mapped[$key][] = $value;
    }
    return $mapped;
}

// Fungsi Reducer: menjumlahkan nilai per kategori
function reducer($mappedData) {
    $reduced = [];
    foreach ($mappedData as $key => $values) {
        $reduced[$key] = array_sum($values);
    }
    return $reduced;
}

// Simulasi MapReduce
$mappedData = mapper($transactions);
$reducedData = reducer($mappedData);

echo "Total Penjualan per Kategori:\n";
foreach ($reducedData as $kategori => $total) {
    echo $kategori . ":" . $total . "\n";
}

/**
 * Analisis:
 * - Mapper: Proses  $O(n)$  pada jumlah transaksi.
 * - Reducer: Proses  $O(m)$  pada jumlah kategori unik.
 * Total simulasi MapReduce memungkinkan pemrosesan data besar
 * dengan distribusi kerja.
 */
```

?>



EXPLORER

SDA-PRAKTIKUM-14

- 1.php
- 2.php
- 3.php
- 4.php
- 5.php

OUTLINE

3.php

```
1 <?php
3 /**
5  * Simulasi sederhana MapReduce untuk menghitung total
6  * penjualan per kategori.
7  */
8
9  // Simulasi data transaksi (kategori, nilai penjualan)
10 $transactions = [
11     ['kategori' => 'Elektronik', 'nilai' => 100],
12     ['kategori' => 'Pakaian', 'nilai' => 50],
13     ['kategori' => 'Elektronik', 'nilai' => 200],
14     ['kategori' => 'Makanan', 'nilai' => 30],
15     ['kategori' => 'Pakaian', 'nilai' => 70],
16     ['kategori' => 'Makanan', 'nilai' => 45],
17 ];
18
19 // Fungsi Mapper: mengelompokkan data per kategori
20 function mapper($transactions) {
21     $mapped = [];
22     foreach ($transactions as $trans) {
23         $key = $trans['kategori'];
24         $value = $trans['nilai'];
25         if (!isset($mapped[$key])) {
26             $mapped[$key] = [];
27         }
28         $mapped[$key][] = $value;
29     }
30     return $mapped;
31 }
32
33 // Fungsi Reducer: menjumlahkan nilai per kategori
34 function reducer($mappedData) {
35     $reduced = [];
36     foreach ($mappedData as $key => $values) {
37         $reduced[$key] = array_sum($values);
38     }
39     return $reduced;
40 }
41
42 // Simulasi MapReduce
43 $mappedData = mapper($transactions);
44 $reducedData = reducer($mappedData);
45
46 echo "Total Penjualan per Kategori:\n";
47 foreach ($reducedData as $kategori => $total) {
48     echo $kategori . ":" . $total . "\n";
49 }
50
51 /**
52  * Analisis:
```

PORTS

GITLENS

TERMINAL

zsh

```
➜ sda-praktikum-14 git:(main) ls -la
total 40
drwxr-xr-x  8 falmesino  staff   256 Feb  8 02:36 .
drwxr-xr-x 38 falmesino  staff  1216 Feb  7 22:45 ..
drwxr-xr-x 12 falmesino  staff   384 Feb  8 03:07 .git
-rw-r--r--  1 falmesino  staff  1832 Feb  8 00:38 1.php
-rw-r--r--  1 falmesino  staff  3728 Feb  8 02:13 2.php
-rw-r--r--  1 falmesino  staff  1557 Feb  8 02:20 3.php
-rw-r--r--  1 falmesino  staff  2728 Feb  8 03:07 4.php
-rw-r--r--  1 falmesino  staff  1811 Feb  8 02:45 5.php

➜ sda-praktikum-14 git:(main) valet php 3.php
Total Penjualan per Kategori:
Elektronik:300
Pakaian:120
Makanan:75

➜ sda-praktikum-14 git:(main)
```

Falmesino Abdul Hamid (35 minutes ago) Ln 19, Col 18 Spaces: 2 UTF-8 LF {} PHP Colorize Colorize: 0 variables Prettier



<?php

```
/**
 * 231232028 - Falmesino Abdul Hamid
 * Contoh sederhana Apriori untuk frequent 1-itemset dan 2-itemset
 */

// Data transaksi: setiap transaksi merupakan array item
$transactions = [
    ['susu', 'roti', 'keju'],
    ['roti', 'keju'],
    ['susu', 'roti'],
    ['roti', 'keju'],
    ['susu', 'keju']
];
$minSupport = 0.6;
$totalTransactions = count($transactions);

// Fungsi untuk menghitung support suatu itemset
function countSupport($transactions, $itemset) {
    $count = 0;
    foreach ($transactions as $trans) {
        if (count(array_intersect($trans, $itemset)) == count($itemset)) {
            $count++;
        }
    }
    return $count / count($transactions);
}

// Mendapatkan frequent 1-itemset
function apriori1itemset($transactions, $minSupport) {
    $itemCounts = [];
    foreach ($transactions as $trans) {
        foreach ($trans as $item) {
            if (!isset($itemCounts[$item])) {
                $itemCounts[$item] = 0;
            }
            $itemCounts[$item]++;
        }
    }
    $frequent1 = [];

    foreach ($itemCounts as $item => $count) {
        $support = $count / count($transactions);
        if ($support >= $minSupport) {
            $frequent1[$item] = $support; // Gunakan string key
        }
    }
    return $frequent1;
}

// Mendapatkan kandidat 2-itemset dari frequent 1-itemset
function generateCandidates($frequent1) {
    $items = array_keys($frequent1); // Ambil item dari frequent1
    $candidates = [];
    $n = count($items);
    for ($i = 0; $i < $n; $i++) {
        for ($j = $i + 1; $j < $n; $j++) {
            $candidate = [$items[$i], $items[$j]];
            sort($candidate); // Urutkan untuk konsistensi
            $candidates[] = $candidate;
        }
    }
    return $candidates;
}

// Evaluasi frequent 2-itemset
function apriori2Itemset($transactions, $candidates, $minSupport) {
    $frequent2 = [];
    foreach ($candidates as $candidate) {
        $support = countSupport($transactions, $candidate);
        if ($support >= $minSupport) {
            $frequent2[implode(',', $candidate)] = $support; // Assign ke frequent2
        }
    }
    return $frequent2;
}

$frequent1 = apriori1itemset($transactions, $minSupport);
$candidates2 = generateCandidates($frequent1);
$frequent2 = apriori2Itemset($transactions, $candidates2, $minSupport);

echo "Frequent 1-itemsets:\n";
print_r($frequent1);
echo "Frequent 2-itemsets:\n";
print_r($frequent2);

/**
 * Analisis:
 * - Proses frequent 1-itemset:  $O(n * k)$  dengan n transaksi dan k item
 *   rata-rata per transaksi.
 * - Proses kandidat 2-itemset:  $O(m^2)$  dengan m item unik.
 * - Support dihitung untuk tiap kandidat dengan proses iteratif.
 */
```

?>



EXPLORER

SDA-PRAKTIKUM-14

- 1.php
- 2.php
- 3.php
- 4.php
- 5.php

4.php

```
1 <?php
3 /**
53 function generateCandidates($frequent1) {
63     }
64     return $candidates;
65 }
66
67 // Evaluasi frequent 2-itemset
68 function apriori2Itemset($transactions, $candidates, $minSupport) {
69     $frequent2 = [];
70     foreach ($candidates as $candidate) {
71         $support = countSupport($transactions, $candidate);
72         if ($support >= $minSupport) {
73             $frequent2[implode(',', $candidate)] = $support; // Assign ke frequent2
74         }
75     }
76     return $frequent2;
77 }
78
79 $frequent1 = apriori1itemset($transactions, $minSupport);
80 $candidates2 = generateCandidates($frequent1);
81 $frequent2 = apriori2Itemset($transactions, $candidates2, $minSupport);
82
83 echo "Frequent 1-itemsets:\n";
84 print_r($frequent1);
85 echo "Frequent 2-itemsets:\n";
86 print_r($frequent2);
87
88 /**
89  * Analisis:
90  * - Proses frequent 1-itemset: O(n * k) dengan n transaksi dan k item
91  *   rata-rata per transaksi.
92  * - Proses kandidate 2-itemset: O(m^2) dengan m item unik.
93  * - Support dihitung untuk tiap kandidat dengan proses iteratif.
94  */
95 ?>
```

PORTS

GITLENS

TERMINAL

```
➜ sda-praktikum-14 git:(main) ls -la
total 40
drwxr-xr-x  8 falmesino  staff   256 Feb  8 02:36 .
drwxr-xr-x 38 falmesino  staff  1216 Feb  7 22:45 ..
drwxr-xr-x 12 falmesino  staff   384 Feb  8 03:07 .git
-rw-r--r--  1 falmesino  staff  1832 Feb  8 00:38 1.php
-rw-r--r--  1 falmesino  staff  3728 Feb  8 02:13 2.php
-rw-r--r--  1 falmesino  staff  1557 Feb  8 02:20 3.php
-rw-r--r--  1 falmesino  staff  2728 Feb  8 03:07 4.php
-rw-r--r--  1 falmesino  staff   1811 Feb  8 02:45 5.php

➜ sda-praktikum-14 git:(main) valet php 4.php
Frequent 1-itemsets:
Array
(
    [susu] => 0.6
    [roti] => 0.8
    [keju] => 0.8
)
Frequent 2-itemsets:
Array
(
    [keju,roti] => 0.6
)

➜ sda-praktikum-14 git:(main)
```

main

Launchpad

0 0

You, 4 minutes ago

Falmesino Abdul Hamid (4 minutes ago)

Ln 71, Col 13

Spaces: 2

UTF-8

LF

{ } PHP

Colorize

Colorize: 0 variables

Prettier



<?php

```
/**
 * 231232028 - Falmesino Abdul Hamid
 * Contoh program untuk mendeteksi anomali berdasarkan EMA:
 */

/**
 * Fungsi untuk menghitung Exponential Moving Average (EMA)
 * dan mendeteksi anomali jika nilai sensor menyimpang terlalu jauh.
 */

function detectAnomaliesEMA($data, $alpha = 0.2, $threshold = 2.0) {
    $ema = $data[0]; // inisialisasi dengan data pertama
    $anomalies = [];
    // Simpan nilai EMA untuk analisis standar deviasi sederhana
    $emaValues = [$ema];

    // Hitung EMA untuk setiap data
    for ($i = 1; $i < count($data); $i++) {
        $ema = $alpha * $data[$i] + (1 - $alpha) * $ema;
        $emaValues[] = $ema;
    }

    // Hitung deviasi sederhana: rata-rata absolute difference antara
    // data dan EMA
    $sumDiff = 0;
    foreach ($data as $i => $value) {
        $sumDiff += abs($value - $emaValues[$i]);
    }
    $meanDiff = $sumDiff / count($data);

    // Deteksi anomali: jika deviasi absolute melebihi threshold * meanDiff
    foreach ($data as $i => $value) {
        if (abs($value - $emaValues[$i]) > $threshold * $meanDiff) {
            $anomalies[$i] = $value;
        }
    }
    return $anomalies;
}

// Contoh data sensor (misalnya, suhu) secara real-time
$sensorData = [22.5, 22.7, 22.6, 22.8, 23.0, 23.1, 22.9, 23.2, 25.0, 23.0, 22.8, 22.7, 22.5];

$anomalies = detectAnomaliesEMA($sensorData, 0.3, 2.0);

echo "Data Sensor:\n";
print_r($sensorData);
echo "\nAnomali Teridentifikasi (indeks => nilai):\n";
print_r($anomalies);

/**
 * Analisis:
 * - Setiap data diproses sekali: O(n).
 * - Ruang yang digunakan konstan (O(1) tambahan, meskipun array emaValues
 *   O(n) dapat dioptimalkan jika hanya nilai terakhir yang disimpan).
 * - Cocok untuk aplikasi IoT dengan data stream secara real-time.
 */
```

?>



The image shows a screenshot of a Visual Studio Code editor interface. The Explorer panel on the left shows a project named 'SDA-PRAKTIKUM-14' with five PHP files: 1.php, 2.php, 3.php, 4.php, and 5.php. File 5.php is selected and open in the main editor. The code in 5.php implements an Exponential Moving Average (EMA) function to detect anomalies in real-time sensor data. It calculates the absolute difference between the current value and the EMA, and if it exceeds a threshold (2.0), it flags it as an anomaly. The script also includes a comment block explaining the analysis complexity. The Output panel on the right shows the terminal output of the script, which displays the sensor data and identifies an anomaly at index 8 with a value of 25. The status bar at the bottom indicates the current file is 'main', the encoding is 'UTF-8', and the language is 'PHP'.