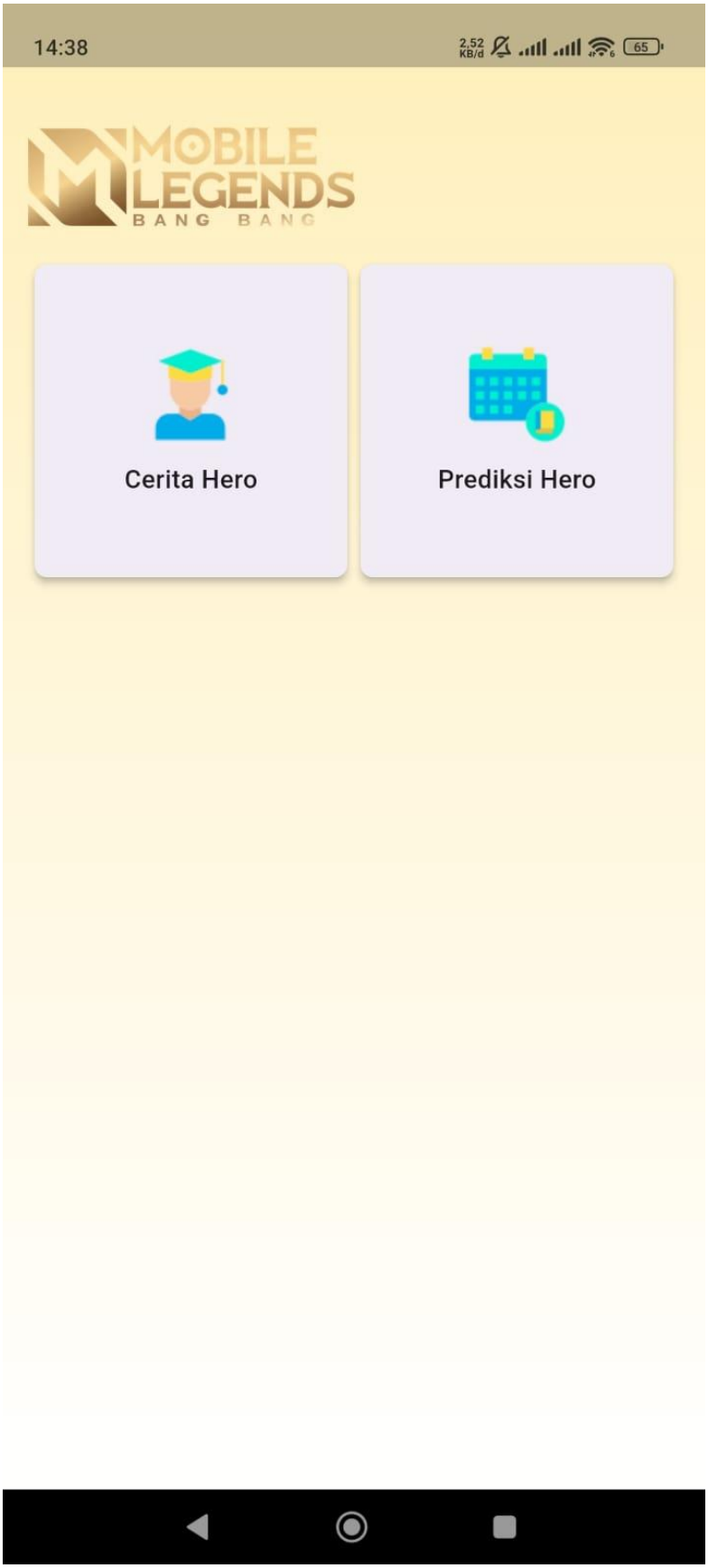
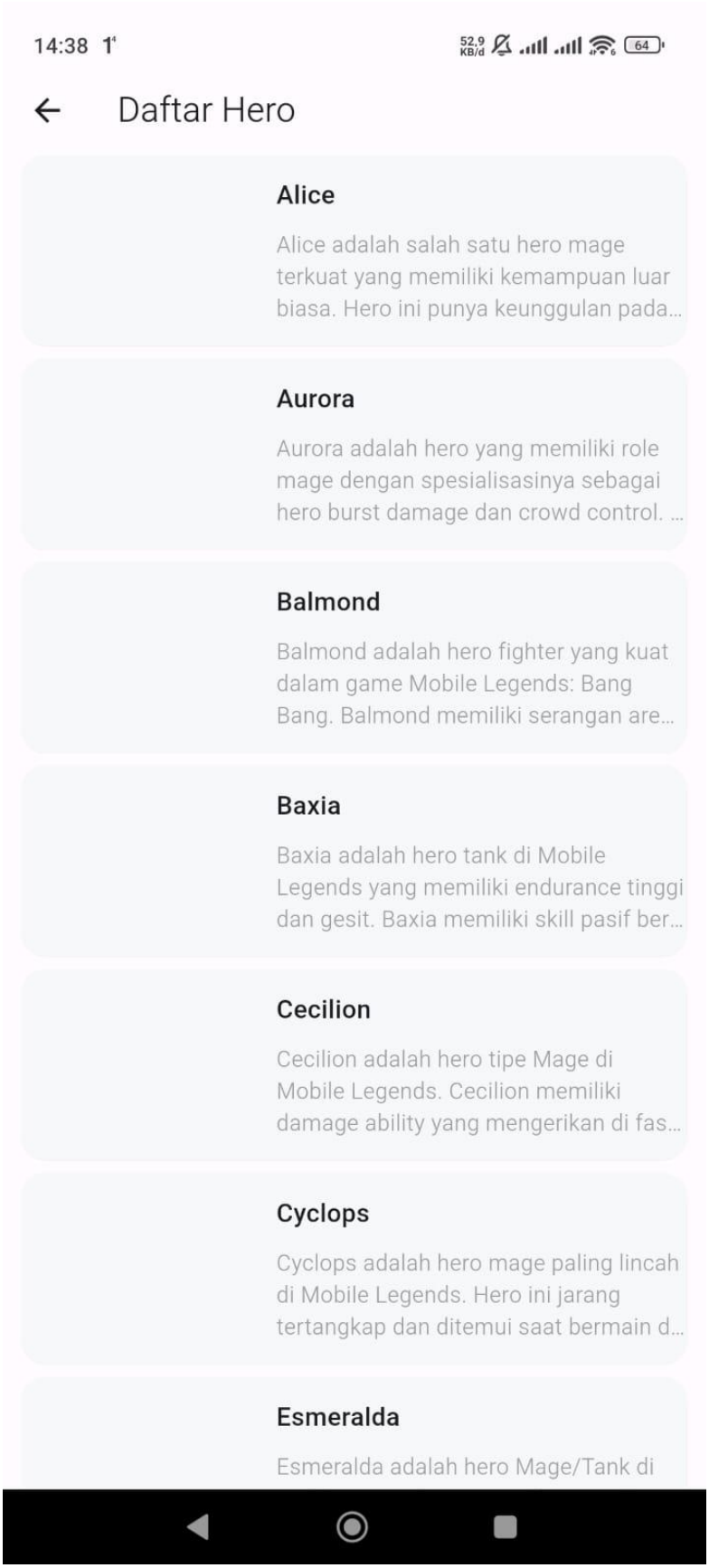
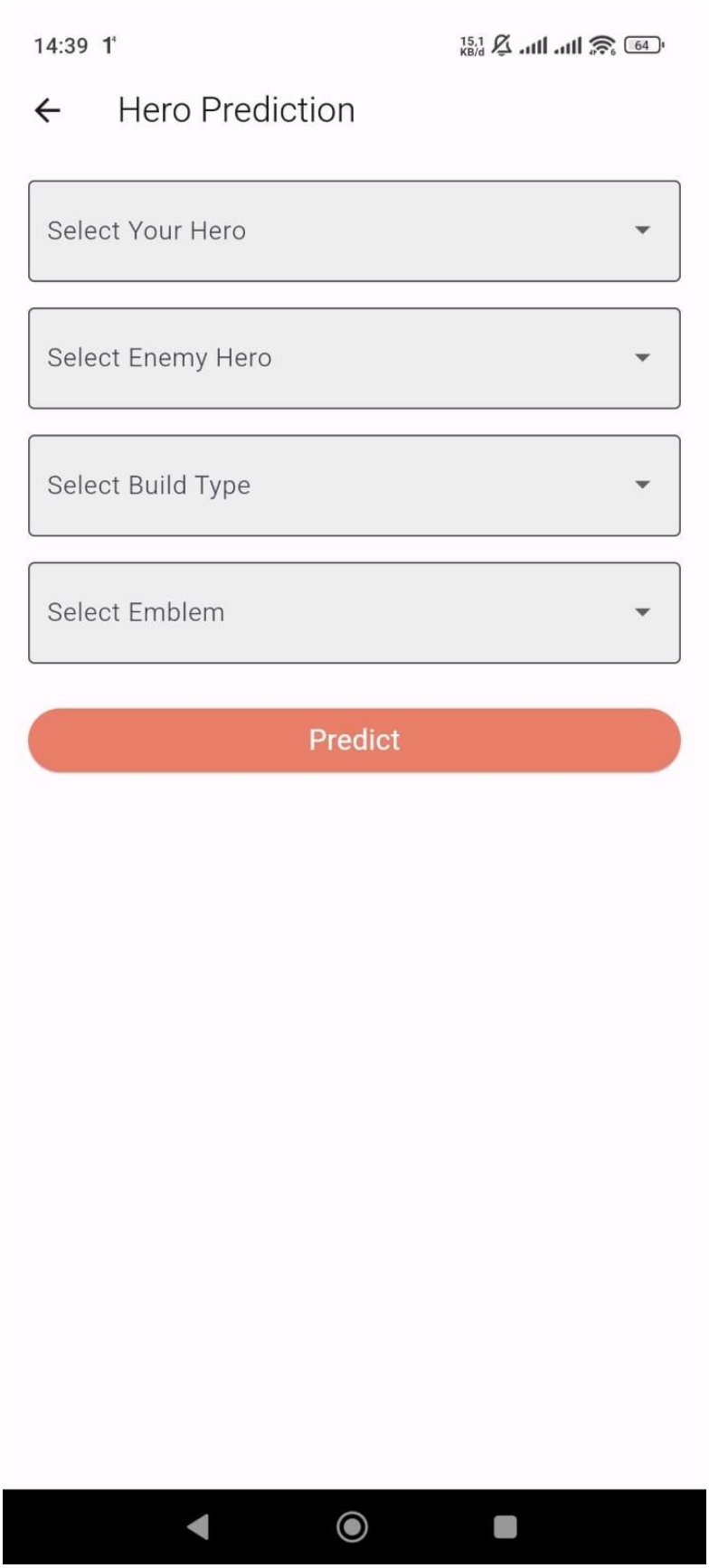
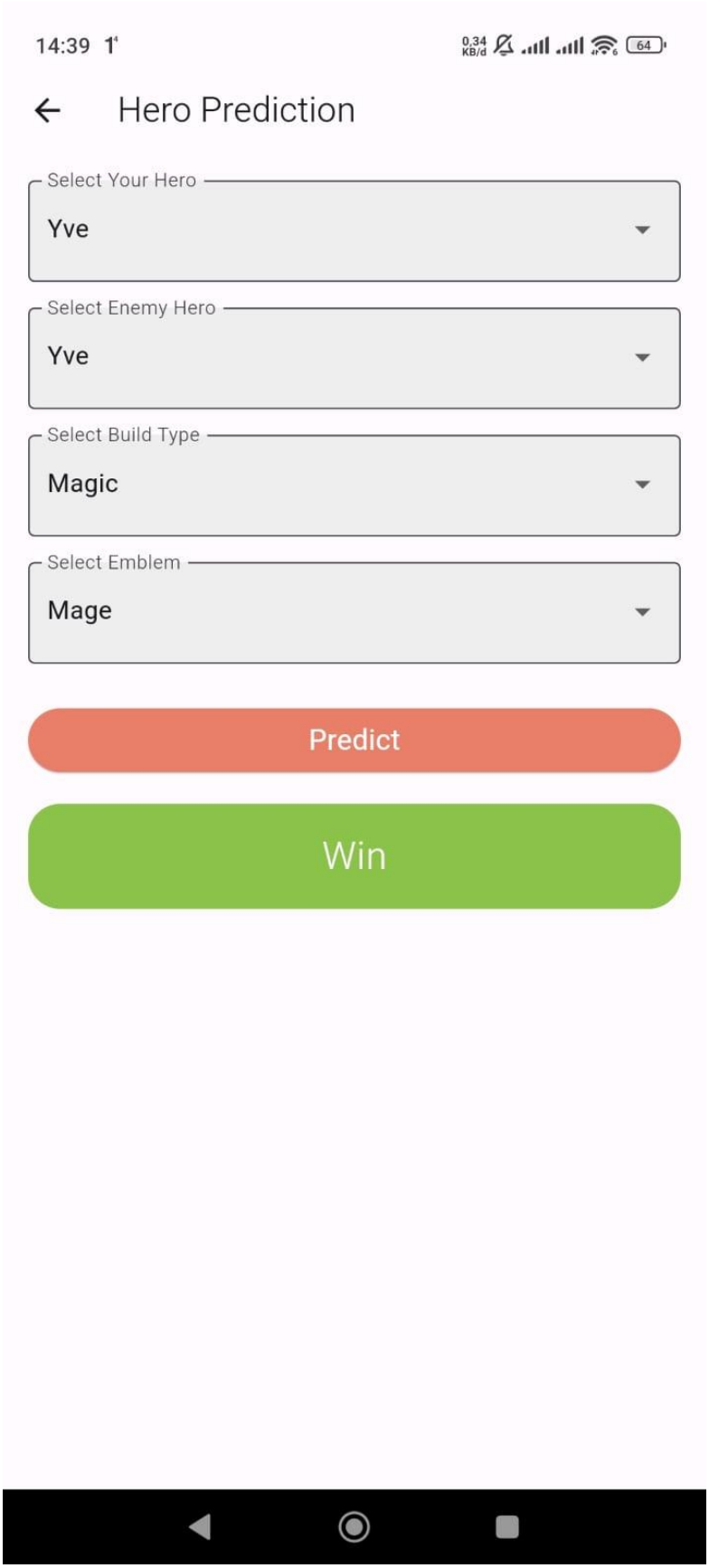
LAMPIRAN

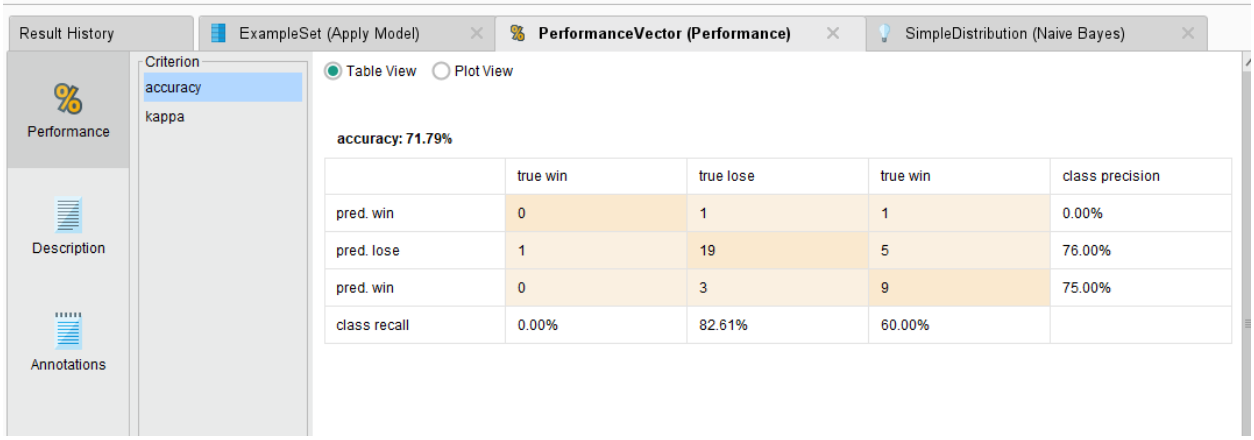

14:37
0,28
KB/d
65
MOBILE
LEGENDS
BANG BANG

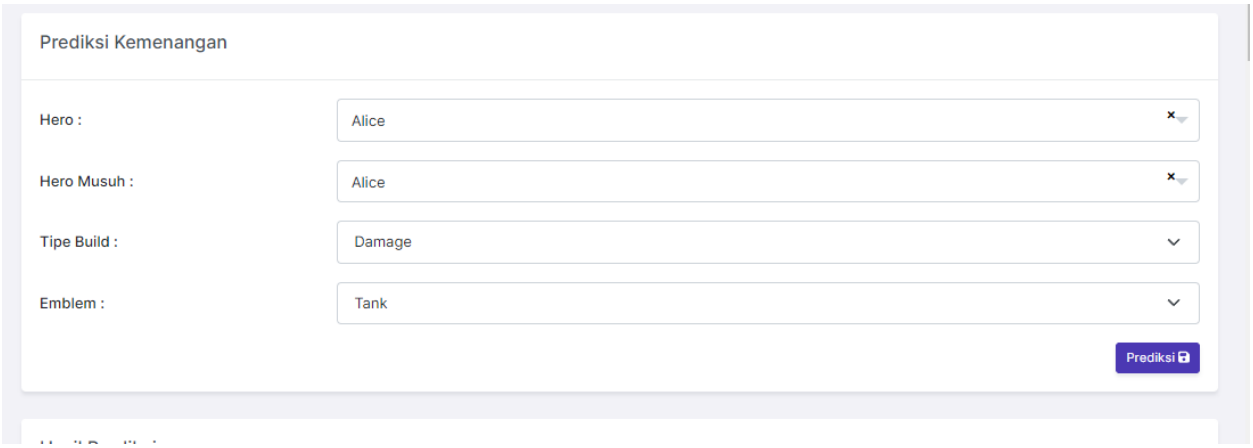

14:38
MOBILE LEGENDS
BANG BANG
Cerita Hero
Prediksi Hero

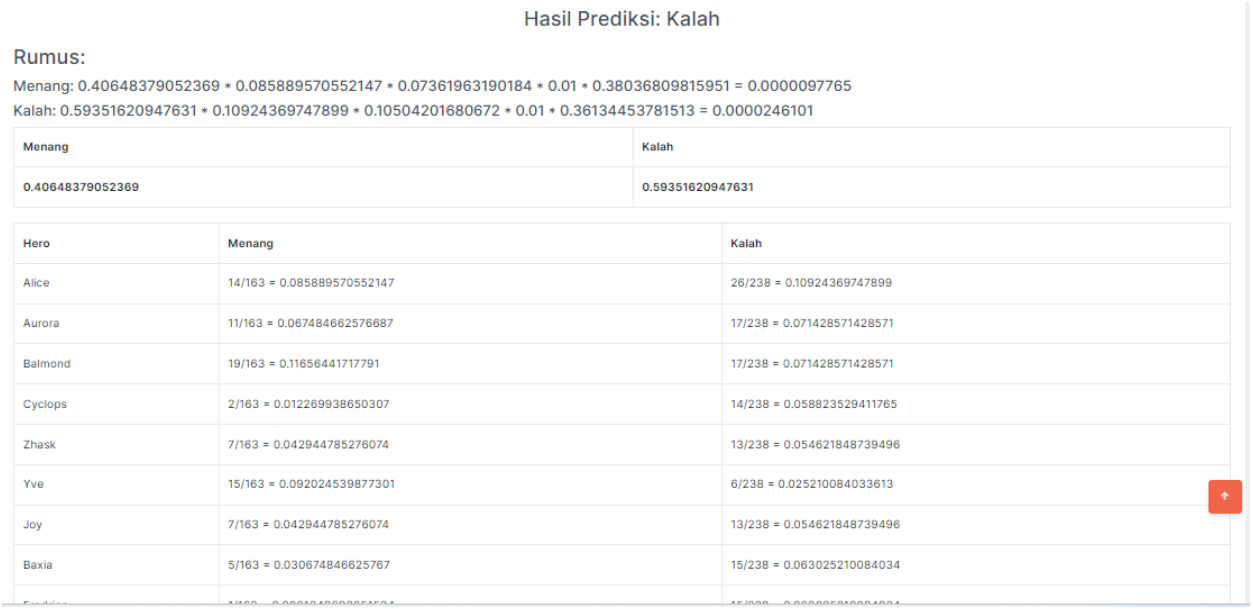

14:38
Daftar Hero
Alice
Alice adalah salah satu hero mage terkuat yang memiliki kemampuan luar biasa. Hero ini punya keunggulan pada...
Aurora
Aurora adalah hero yang memiliki role mage dengan spesialisasinya sebagai hero burst damage dan crowd control. ...
Balmond
Balmond adalah hero fighter yang kuat dalam game Mobile Legends: Bang Bang. Balmond memiliki serangan are...
Baxia
Baxia adalah hero tank di Mobile Legends yang memiliki endurance tinggi dan gesit. Baxia memiliki skill pasif ber...
Cecilion
Cecilion adalah hero tipe Mage di Mobile Legends. Cecilion memiliki damage ability yang mengerikan di fas...
Cyclops
Cyclops adalah hero mage paling lincah di Mobile Legends. Hero ini jarang tertangkap dan ditemui saat bermain d...
Esmeralda
Esmeralda adalah hero Mage/Tank di


14:39
Hero Prediction
Select Your Hero
Select Enemy Hero
Select Build Type
Select Emblem
Predict


14:39
Hero Prediction
Select Your Hero
Yve
Select Enemy Hero
Yve
Select Build Type
Magic
Select Emblem
Mage
Predict
Win


Result History
ExampleSet (Apply Model)
PerformanceVector (Performance)
SimpleDistribution (Naive Bayes)
Performance
Description
Annotations
Criterion
accuracy
kappa
Table View
Plot View
accuracy: 71.79%
true win
true lose
true win
class precision
pred. win
0
1
1
0.00%
pred. lose
1
19
5
76.00%
pred. win
0
3
9
75.00%
class recall
0.00%
82.61%
60.00%


Prediksi Kemenangan
Hero :
Alice
Hero Musuh :
Alice
Tipe Build :
Damage
Emblem :
Tank
Prediksi


Hasil Prediksi: Kalah
Rumus:
Menang: 0.40648379052369 * 0.085889570552147 * 0.07361963190184 * 0.01 * 0.38036809815951 = 0.0000097765
Kalah: 0.59351620947631 * 0.10924369747899 * 0.10504201680672 * 0.01 * 0.36134453781513 = 0.0000246101
Menang
Kalah
0.40648379052369
0.59351620947631
Hero
Menang
Kalah
Alice
14/163 = 0.085889570552147
26/238 = 0.10924369747899
Aurora
11/163 = 0.067484662576687
17/238 = 0.071428571428571
Balmond
19/163 = 0.11656441717791
17/238 = 0.071428571428571
Cyclops
2/163 = 0.012269938650307
14/238 = 0.058823529411765
Zhask
7/163 = 0.042944785276074
13/238 = 0.054621848739496
Yve
15/163 = 0.092024539877301
6/238 = 0.025210084033613
Joy
7/163 = 0.042944785276074
13/238 = 0.054621848739496
Baxia
5/163 = 0.030674846625767
15/238 = 0.063025210084034

```php
namespace App\Helpers;

class NaiveBayesClassifier {
  private $rumusPredict = [];
  private $trainingData = [];
  private $classCounts = [];
  private $featureCounts = [];
  private $tableCounts = [];

  public function __construct($trainingData) {
    $this->trainingData = $trainingData;
    $this->calculateProbabilities();
  }

  public function calculateProbabilities() {
    $totalData = count($this->trainingData);

    // Menghitung jumlah masing-masing kelas
    foreach ($this->trainingData as $data) {
      $class = $data['hasil'];
      if (!isset($this->classCounts[$class])) {
        $this->classCounts[$class] = 0;
      }
      $this->classCounts[$class]++;
    }

    // Menghitung jumlah masing-masing fitur untuk setiap kelas
    foreach ($this->trainingData as $data) {
      foreach ($data as $key => $value) {
        if ($key !== 'hasil') {
          $class = $data['hasil'];
          if (!isset($this->featureCounts[$class][$key][$value])) {
            $this->featureCounts[$class][$key][$value] = 0;
          }
          $this->featureCounts[$class][$key][$value]++;
        }
      }
    }

    // Menghitung probabilitas
    foreach ($this->classCounts as $class => $count) {
      foreach ($this->featureCounts[$class] as $featureKey => $featureValues)
{
        foreach ($featureValues as $value => $valueCount) {
//          $this->featureCounts[$class][$featureKey][$value] = $this-
>featureCounts[$class][$featureKey][$value]."/{$count} = ". $this-
>featureCounts[$class][$featureKey][$value] / $count;
```

```
          $this->tableCounts[$class][$featureKey][$value] = $this-
>featureCounts[$class][$featureKey][$value]."/{$count} = ". $this-
>featureCounts[$class][$featureKey][$value] / $count;
//          dd($this->featureCounts[$class][$featureKey][$value] = $value++);
          $this->featureCounts[$class][$featureKey][$value] /= $count;
//          dd($this->featureCounts[$class][$featureKey][$value]);
        }
      }
      $this->classCounts[$class] /= $totalData;
    }
  }

  public function predict($data) {
    $classProbabilities = []; // Menyimpan probabilitas untuk setiap kelas
    $predictedClass = '';
    $maxProbability = -1;
    $rumus = '';

    foreach ($this->classCounts as $class => $classProbability) {
      $probability = $classProbability; // Menginisialisasi probabilitas kelas
      $rumus .= $classProbability." * "; // Menginisialisasi probabilitas
kelas
      // Perhitungan probabilitas fitur diberikan kelas
      $keysArray = array_keys($data);
      $lastKey = end($keysArray);
      $kali = '';
      foreach ($data as $key => $value) {
        if ($key !== 'hasil') {
          // Menggunakan probabilitas fitur diberikan kelas (P(xi|C))
          // Memeriksa jika fitur ada dalam data pelatihan
          if (isset($this->featureCounts[$class][$key][$value])) {
            // Perhitungan probabilitas fitur diberikan kelas
            $probability *= $this->featureCounts[$class][$key][$value];

            if ($key !== $lastKey) {
              $kali =  " * ";
            }

            $rumus .= $this->featureCounts[$class][$key][$value] .$kali;
          } else {
            // Handle jika fitur tidak ada di data pelatihan menggunakan
Laplace smoothing
            // Ini hanyalah contoh pendekatan, dalam kasus nyata mungkin
diperlukan pendekatan yang lebih canggih
            $probability *= 0.01; // Contoh: Laplace smoothing
            if ($key !== $lastKey) {
              $kali =  " * ";
            }
```

```
            $rumus .= "0.01 {$kali}";
          }
        }
        $kali = '';
      }
      $this->rumusPredict[$class] = $rumus ." = ".number_format($probability,
10);
      $rumus = '';

      // Menyimpan nilai probabilitas untuk setiap kelas
      $classProbabilities[$class] = $probability;

      // Memilih kelas dengan probabilitas tertinggi sebagai kelas prediksi
      if ($probability > $maxProbability) {
        $maxProbability = $probability;
        $predictedClass = $class;
      }
    }
    unset($this->trainingData);
    return $predictedClass;
  }

  public function getTableCounts() {
    // Menampilkan jumlah masing-masing fitur untuk setiap kelas

    $organizedData = [];
    foreach ($this->tableCounts as $outcome => $categories) {
      foreach ($categories as $categoryName => $categoryValues) {
        foreach ($categoryValues as $itemName => $itemValue) {
          $organizedData[$categoryName][$itemName][$outcome] = $itemValue;
        }

        // Menambahkan nilai default 0 jika item tidak ada dalam array
'Menang' atau 'Kalah'
        $allItems = array_keys($this->tableCounts['Menang'][$categoryName] +
$this->tableCounts['Kalah'][$categoryName]);
        $missingItems = array_diff($allItems,
array_keys($categories[$categoryName]));

        foreach ($missingItems as $missingItem) {
          $organizedData[$categoryName][$missingItem][$outcome] = '0';
        }
      }
    }
    return $organizedData;
  }

  public function getRumusPredict(){
```

```
    return $this->rumusPredict;
  }

  public function getClassCount(){
    return $this->classCounts;
  }
}
```