APPENDIX

CODE TO CALCULTING USING DECISION TREE C45

```
<?php

namespace App\Services;

use Illuminate\Database\Eloquent\Collection;
use Illuminate\Database\Eloquent\Builder;

use App\Models\Dataset\Dataset;
use App\Models\Summary\Summary;
use App\Models\Summary\SummaryDetail;
use App\Models\Summary\SummaryDetailDataset;
use App\Models\Summary\ResultComparison;
class C45
{
    protected $datasets;
    protected $gender;
    protected $age;
    protected $attribute;
    protected $node;

    public function __construct(string $gender, string $age, string $attribute) {
        $this->datasets = Dataset::query();
        $this->datasets->where('gender', $gender)->where('age', $age);

        $this->gender = $gender;
        $this->age = $age;
        $this->attribute = $attribute;
        $this->node = 1;

        ResultComparison::where('type', 'c45')->delete();
    }

    public function processCount() {
        $processId   = [];
        $exceptQuery = [];
        $totalRoot   = null;

        $counter = 0;
        while(true) {
            if ($totalRoot !== null && $counter >= $totalRoot) {
                break;
            }

            $mainProcess       =       $this->_mainProcess($processId, $exceptQuery, $totalRoot);
            $processId   = array_merge($processId, $mainProcess->pid);
            $exceptQuery   =   array_merge($exceptQuery,   $mainProcess->exception);
            $totalRoot   = $mainProcess->total_root;
```

```
73.
74.             $counter++;
75.         }
76.
77.         return $processId;
78.     }
79.
80.     private function _mainProcess($processId = [], $exceptQuery = [],
    $totalRoot = null) {
81.         $datasets = (clone $this->datasets)
82.             ->when(count($exceptQuery)   >   0,   function($subQuery)
    use($exceptQuery) {
83.                 return               $this->_scopeException($subQuery,
    $exceptQuery);
84.             })->get();
85.
86.         $summary = Summary::create([
87.             'pid'  => $this->_incrementProcessId(),
88.             'node' => $this->_incrementNode($processId)
89.         ]);
90.
91.         $processId[] = $summary->pid;
92.
93.         /** Calculate attribute total #1 */
94.         $this->_calculateAttributeTotal($datasets, $summary->id);
95.
96.         /** Calculate attribute primary #1 */
97.         $candidatePrimaryAttributes     =     $datasets->pluck($this-
    >attribute)->unique()->toArray();
98.         $this->_calculateAttributePrimary($datasets,
    $candidatePrimaryAttributes, $summary->id);
99.
100.        $secondaryAttributeNames = [
101.            'fav_drink'=> 'fav_drink_flavour',
102.            'fav_food' => 'fav_food_flavour'
103.        ];
104.
105.        /** Calculate attribute secondary #1 */
106.        $candidateSecondaryAttributes          =          $datasets-
    >pluck($secondaryAttributeNames[$this->attribute])->unique()->toArray();
107.        $this->_calculateAttributeSecondary($datasets,
    $candidateSecondaryAttributes, $summary->id);
108.
109.        /** Looking for highest gain #1 */
110.        $highestGainPrimary    =    SummaryDetail::where('summary_id',
    $summary->id)
111.            ->where('attribute', '!=', 'Total')
112.            ->orderBy('gain', 'desc')
113.            ->first();
114.
115.        /** Looking for highest entrophy #1 */
116.        $highestEntrophyPrimary                                        =
    SummaryDetailDataset::where('summary_detail_id',    $highestGainPrimary-
    >id)
117.            ->orderBy('entrophy', 'desc')
118.            ->first();
```

```
119.
120.         $attributeType = $highestEntrophyPrimary->type;
121.         $columnValue   = $highestEntrophyPrimary->attribute;
122.
123.         $columnName = [
124.             'fav_drink' => [
125.                 'primary'   => 'fav_drink',
126.                 'secondary' => 'fav_drink_flavour'
127.             ],
128.             'fav_food' => [
129.                 'primary'   => 'fav_food',
130.                 'secondary' => 'fav_food_flavour'
131.             ]
132.         ];
133.
134.         $datasets         =         $datasets->where($columnName[$this-
   >attribute][$attributeType], $columnValue);
135.
136.         $summary = Summary::create([
137.             'pid'  => $this->_incrementProcessId(),
138.             'node' => $this->_incrementNode($processId)
139.         ]);
140.
141.         $processId[] = $summary->pid;
142.
143.         /** Calculate attribute total #2 */
144.         $this->_calculateAttributeTotal($datasets, $summary->id);
145.
146.         /** Calculate attribute primary #2 */
147.         $candidatePrimaryAttributes      =      $datasets->pluck($this-
   >attribute)->unique()->toArray();
148.         $this->_calculateAttributePrimary($datasets,
   $candidatePrimaryAttributes, $summary->id);
149.
150.         $secondaryAttributeNames = [
151.             'fav_drink'=> 'fav_drink_flavour',
152.             'fav_food' => 'fav_food_flavour'
153.         ];
154.
155.         /** Calculate attribute secondary #2 */
156.         $candidateSecondaryAttributes           =           $datasets-
   >pluck($secondaryAttributeNames[$this->attribute])->unique()->toArray();
157.         $this->_calculateAttributeSecondary($datasets,
   $candidateSecondaryAttributes, $summary->id);
158.
159.         /** Looking for highest gain #2 */
160.         $highestGainSecondary   =   SummaryDetail::where('summary_id',
   $summary->id)
161.             ->where('attribute', '!=', 'Total')
162.             ->orderBy('gain', 'desc')
163.             ->first();
164.
165.         /** Looking for highest entrophy #2 */
166.         $highestEntrophySecondary                                    =
   SummaryDetailDataset::where('summary_detail_id',  $highestGainSecondary-
   >id)
```

```
167.             ->orderBy('entrophy', 'desc')
168.             ->first();
169.
170.         /** Set recommendation */
171.         $recommendation = $this->_createRecommendation($summary->id);
172.         $summary = Summary::create([
173.             'pid'    => $this->_incrementProcessId(),
174.             'node'   => $this->_incrementNode($processId),
175.             'result' => 'Result : ' . $recommendation['primary'] . ' '
    . $recommendation['secondary']
176.         ]);
177.
178.         ResultComparison::create([
179.             'type'      => 'c45',
180.             'gender'    => $this->gender,
181.             'age'       => $this->age,
182.             'menu'      => $recommendation['primary'],
183.             'flavour'   => $recommendation['secondary'],
184.             'label'     => 'Rekomendasi'
185.         ]);
186.
187.         $processId[] = $summary->pid;
188.
189.
190.         /** Set value total root for first process */
191.         $countHighestGain                                                =
    SummaryDetailDataset::where('summary_detail_id',    $highestGainPrimary-
    >id)->count();
192.         if ($totalRoot === null) $totalRoot = $countHighestGain;
193.
194.         /** Set value query exception */
195.         $exceptQuery[] = $highestEntrophyPrimary->attribute;
196.
197.         return (object) [
198.             'pid'        => $processId,
199.             'exception'  => $exceptQuery,
200.             'total_root' => $totalRoot
201.         ];
202.     }
203.
204.     private function _calculateAttributeTotal(Collection $datasets, int
    $summaryId) {
205.         $total = $datasets->count();
206.         $attribute = $this->attribute;
207.
208.         $totalTrue   =   $datasets->when($attribute   ==   'fav_drink',
    function($items) {
209.             return $items->where('recommend_drink', 1);
210.         })
211.         ->when($attribute == 'fav_food', function($items) {
212.             return $items->where('recommend_food', 1);
213.         })
214.         ->count();
215.
216.         $totalFalse   =   $datasets->when($attribute   ==   'fav_drink',
    function($items) {
```

```
217.             return $items->where('recommend_drink', 0);
218.         })
219.         ->when($attribute == 'fav_food', function($items) {
220.             return $items->where('recommend_food', 0);
221.         })
222.         ->count();
223.
224.         $entrophy   =   $this->_calculateEntrophy($total,   $totalTrue,
   $totalFalse);
225.
226.         $summaryDetail = SummaryDetail::create([
227.             'summary_id'    => $summaryId,
228.             'attribute'     => 'Total',
229.             'total'         => $total,
230.             'total_true'    => $totalTrue,
231.             'total_false'   => $totalFalse,
232.             'entrophy'      => $entrophy
233.         ]);
234.     }
235.
236.     private function _calculateAttributePrimary(Collection $datasets,
   array $candidateAttributes, int $summaryId) {
237.         $columnName = $this->attribute;
238.
239.         $attributeName = [
240.             'fav_drink'=> 'Minuman Favorite',
241.             'fav_food' => 'Makanan Favorite'
242.         ];
243.
244.         $summaryDetail = SummaryDetail::create([
245.             'summary_id' => $summaryId,
246.             'attribute'  => $attributeName[$columnName]
247.         ]);
248.
249.         $data = [];
250.         foreach($candidateAttributes as $key => $attribute) {
251.             $data[$key]['total']    =    $datasets->where($columnName,
   $attribute)->count();
252.
253.             $data[$key]['totalTrue']   =   $totalTrue   =   $datasets-
   >where($columnName, $attribute)
254.                 ->when($columnName == 'fav_drink', function($items) {
255.                     return $items->where('recommend_drink', 1);
256.                 })
257.                 ->when($columnName == 'fav_food', function($items) {
258.                     return $items->where('recommend_food', 1);
259.                 })
260.                 ->count();
261.
262.             $data[$key]['totalFalse']   =   $totalFalse   =   $datasets-
   >where($columnName, $attribute)
263.                 ->when($columnName == 'fav_drink', function($items) {
264.                     return $items->where('recommend_drink', 0);
265.                 })
266.                 ->when($columnName == 'fav_food', function($items) {
267.                     return $items->where('recommend_food', 0);
```

```
                })
                ->count();

            $data[$key]['entrophy']             =             $this->_calculateEntrophy($data[$key]['total'],      $data[$key]['totalTrue'], $data[$key]['totalFalse']);

            $summaryDetailDataset = SummaryDetailDataset::create([
                'summary_detail_id' => $summaryDetail->id,
                'type'              => 'primary',
                'attribute'         => $attribute,
                'total'             => $data[$key]['total'],
                'total_true'        => $data[$key]['totalTrue'],
                'total_false'       => $data[$key]['totalFalse'],
                'entrophy'          => $data[$key]['entrophy']
            ]);
        }

        $dataTotal = SummaryDetail::where('summary_id', $summaryId)->where('attribute', 'Total')->first();
        $summaryDetail->gain    =    $this->_calculateGain($dataTotal, $data);
        $summaryDetail->save();
    }

    private function _calculateAttributeSecondary(Collection $datasets, array $candidateSecondaryAttributes, int $summaryId) {
        $attributeName = [
            'fav_drink' => [
                'column'    => 'fav_drink_flavour',
                'attribute' => 'Rasa Minuman'
            ],
            'fav_food'  => [
                'column'    => 'fav_food_flavour',
                'attribute' => 'Rasa Makanan'
            ]
        ];

        $columnName = $attributeName[$this->attribute]['column'];

        $summaryDetail = SummaryDetail::create([
            'summary_id' => $summaryId,
            'attribute'                 =>        $attributeName[$this->attribute]['attribute']
        ]);

        $data = [];
        foreach($candidateSecondaryAttributes as $key => $attribute) {
            $data[$key]['total']    =    $datasets->where($columnName, $attribute)->count();

            $data[$key]['totalTrue']   =   $totalTrue   =   $datasets->where($columnName, $attribute)
                ->when($columnName       ==       'fav_drink_flavour', function($items) {
                    return $items->where('recommend_drink', 1);
```

```
315.                 })
316.                 ->when($columnName        ==        'fav_food_flavour',
   function($items) {
317.                     return $items->where('recommend_food', 1);
318.                 })
319.                 ->count();
320.
321.             $data[$key]['totalFalse']   =   $totalFalse   =   $datasets-
   >where($columnName, $attribute)
322.                 ->when($columnName       ==       'fav_drink_flavour',
   function($items) {
323.                     return $items->where('recommend_drink', 0);
324.                 })
325.                 ->when($columnName        ==        'fav_food_flavour',
   function($items) {
326.                     return $items->where('recommend_food', 0);
327.                 })
328.                 ->count();
329.
330.             $data[$key]['entrophy']              =              $this-
   >_calculateEntrophy($data[$key]['total'],      $data[$key]['totalTrue'],
   $data[$key]['totalFalse']);
331.
332.             $summaryDetailDataset = SummaryDetailDataset::create([
333.                 'summary_detail_id' => $summaryDetail->id,
334.                 'type'              => 'secondary',
335.                 'attribute'         => $attribute,
336.                 'total'             => $data[$key]['total'],
337.                 'total_true'        => $data[$key]['totalTrue'],
338.                 'total_false'       => $data[$key]['totalFalse'],
339.                 'entrophy'          => $data[$key]['entrophy']
340.             ]);
341.         }
342.
343.         $dataTotal  =  SummaryDetail::where('summary_id',  $summaryId)-
   >where('attribute', 'Total')->first();
344.         $summaryDetail->gain    =    $this->_calculateGain($dataTotal,
   $data);
345.         $summaryDetail->save();
346.     }
347.
348.     private function _calculateEntrophy(int $total, int $totalTrue, int
   $totalFalse) {
349.         $entrophy = ((($totalTrue / $total) * -1) * (log(($totalTrue /
   $total), 2))) + ((($totalFalse / $total) * -1) * (log(($totalFalse /
   $total), 2)));
350.         return is_nan($entrophy) ? 0 : $entrophy;
351.     }
352.
353.     private function _calculateGain($dataTotal, $dataEntrophies) {
354.         foreach($dataEntrophies as $dataEntrophie) {
355.             $dataTotal->entrophy    -=    ($dataEntrophie['total']    /
   $dataTotal->total) * $dataEntrophie['entrophy'];
356.         }
357.
358.         return is_nan($dataTotal->entrophy) ? 0 : $dataTotal->entrophy;
```

```
359.     }
360.
361.     private function _createRecommendation(int $summaryId) {
362.         $this->node = $this->node + 0.1;
363.
364.         $highestPrimaryAttribute                                    =
   SummaryDetailDataset::whereHas('summaryDetail',      function($subQuery)
   use($summaryId) {
365.                 $subQuery->where('summary_id', $summaryId);
366.                 $subQuery->whereIn('attribute',  ['Minuman  Favorite',
   'Makanan Favorite']);
367.             })
368.             ->orderBy('entrophy', 'desc')
369.             ->first();
370.
371.         $highestSecondaryAttribute                                  =
   SummaryDetailDataset::whereHas('summaryDetail',      function($subQuery)
   use($summaryId) {
372.                 $subQuery->where('summary_id', $summaryId);
373.                 $subQuery->whereIn('attribute', ['Rasa Minuman', 'Rasa
   Makanan']);
374.             })
375.             ->orderBy('entrophy', 'desc')
376.             ->first();
377.
378.         $attributeName = [
379.             'fav_drink' => 'Menu Minuman',
380.             'fav_food'  => 'Menu Makanan'
381.         ];
382.
383.         // $result = 'Result : ' . $highestPrimaryAttribute->attribute
   . ' ' . $highestSecondaryAttribute->attribute;
384.         $result = ['primary' => $highestPrimaryAttribute->attribute,
   'secondary' => $highestSecondaryAttribute->attribute];
385.         return $result;
386.     }
387.
388.     private function _incrementProcessId() {
389.         $lastRecord = Summary::groupBy('pid')->orderBy('pid', 'desc')-
   >first();
390.         $counter = $lastRecord ? ($lastRecord->pid + 1) : 1;
391.         return $counter;
392.     }
393.
394.     private function _incrementNode(array $pid) {
395.         if (count($pid) > 0) {
396.             $summary  =  Summary::whereIn('pid',  $pid)->orderBy('pid',
   'desc')->first();
397.
398.             return $summary->node + 0.01;
399.         }
400.
401.         return 1.00;
402.     }
403.
404.     private function _scopeException(Builder $query, array $value) {
```

```
405.         return $query->where(function($subQuery) use($value) {
406.             $exceptions     =     ['fav_drink',     'fav_drink_flavour',
   'fav_food', 'fav_food_flavour'];
407.
408.             foreach($exceptions as $column) {
409.                 $subQuery->whereNotIn($column, $value);
410.             }
411.         });
412.     }
}
```

CODE TO CALCULATING USING KNN

```
1. CREATE <?php
2.
3. namespace App\Services;
4.
5. use Illuminate\Database\Eloquent\Collection;
6. use Illuminate\Database\Eloquent\Builder;
7.
8. use App\Models\Dataset\Dataset;
9. use App\Models\Knn\ResultKnn;
10.
11.  class Knn
12.  {
13.      public function processCount(array $input) {
14.
15.          $kValue = $input['k_value'];
16.          $query = Dataset::query();
17.          $query->where('gender',         $input['gender'])->where('age',
   $input['age']);
18.
19.          if ($input['attribute'] == 'fav_drink') {
20.              $query->selectRaw("fav_drink     AS      primary_attribute,
   fav_drink_flavour    AS    secondary_attribute,    COUNT(*)    AS    total,
   SUM(if(recommend_drink     =     '1',     1,     0))     AS     total_true,
   SUM(if(recommend_drink = '0', 1, 0)) AS total_false")
21.                  ->groupBy(['fav_drink', 'fav_drink_flavour'])
22.                  ->orderBy('fav_drink', 'asc')
23.                  ->orderBy('fav_drink_flavour', 'asc');
24.          }
25.          else if ($input['attribute'] == 'fav_food') {
26.
27.
28.              $query->selectRaw("fav_food      AS      primary_attribute,
   fav_food_flavour    AS    secondary_attribute,    COUNT(*)    AS    total,
   SUM(if(recommend_food = '1', 1, 0)) AS total_true, SUM(if(recommend_food
   = '0', 1, 0)) AS total_false")
29.                  ->groupBy(['fav_food', 'fav_food_flavour'])
30.                  ->orderBy('fav_food', 'asc')
31.                  ->orderBy('fav_food_flavour', 'asc');
32.          }
33.
34.          $datasets = $query->get();
35.
36.          $result = [];
```

```
37.          foreach($datasets as $index => $dataset) {
38.              $result[$index]['primary_attribute']      =      $dataset-
   >primary_attribute;
39.              $result[$index]['secondary_attribute']     =     $dataset-
   >secondary_attribute;
40.              $result[$index]['total'] = $dataset->total;
41.              $result[$index]['total_true'] = $dataset->total_true;
42.              $result[$index]['total_false'] = $dataset->total_false;
43.              $result[$index]['label']    =    $this->_addLabel($dataset-
   >total_true, $dataset->total_false);
44.
45.              $input['total'] = isset($input['total']) ? $input['total']
   : $dataset->total;
46.              $input['total_true']   =   isset($input['total_true'])   ?
   $input['total_true'] : $dataset->total_true;
47.              $input['total_false']   =   isset($input['total_false'])   ?
   $input['total_false'] : $dataset->total_false;
48.
49.              $result[$index]['distance'] = $this->_sumDistance(
50.                  $dataset->total,
51.                  $dataset->total_true,
52.                  $dataset->total_false,
53.                  $input['total'],
54.                  $input['total_true'],
55.                  $input['total_false'],
56.              );
57.
58.              $result[$index]['k_value']  = $kValue;
59.              $result[$index]['result_k'] = null;
60.
61.              $result[$index]['created_at'] = now();
62.              $result[$index]['updated_at'] = now();
63.          }
64.
65.          ResultKnn::insert($result);
66.
67.          $resultKnn     =     ResultKnn::orderBy('distance',     'asc')-
   >take($kValue)->get();
68.          foreach($resultKnn as $knn) {
69.              $knn->result_k = $knn->label;
70.              $knn->save();
71.          }
72.
73.          $knn = ResultKnn::orderBy('distance', 'asc')->get();
74.          return [
75.              'data'   => $knn,
76.              'result'    =>    (clone    $knn)->whereNotNull('result_k')-
   >mode('result_k')
77.          ];
78.      }
79.
80.      private function _addLabel($totalTrue, $totalFalse) {
81.          if ($totalTrue > $totalFalse) {
82.              return 'Rekomendasi';
83.          }
84.
```

```
85.           return 'Tidak Rekomendasi';
86.       }
87.
88.       private function _sumDistance($total, $totalTrue, $totalFalse,
   $sampleTotal, $sampleTotalTrue, $sampleTotalFalse) {
89.           return sqrt(pow(($total - $sampleTotal), 2) + pow(($totalTrue -
   $sampleTotalTrue), 2) + pow(($totalFalse - $sampleTotalFalse), 2));
90.       }
91.   }
```

Report #13397899

INTRODUCTION Background At this time there are so many cofee shop that serve a variety of food menus. Because of this, of course, potential visitors will be more selective because of the increasing number of cafe choices. Each cafe itself certainly has a variety of menu choices that will be served to visitors. But from the variety of existing menus, a problem arises that is closely related to the recommendation system. The existing recommendations are deemed ineffective because they only look at the number of menus that are sold the most without paying attention to the age and gender background of the customer. Whereas in a cafe, it must be considered whether the menu ordered is suitable to be enjoyed by a certain age and gender range.
To solve the problem, the menu recommendation system should be further developed. The existing recommendation system should reclassify based on a certain age range and gender. Of course, this aims to simplify the service process when ordering a menu. This final project will use two