Lampiran

Source Code Proses *MIN-MAX NORMALIZATION*

```
class KlasifikasiController extends Controller
{
    private function getMinMaxValues()
    {
        $minMaxValues = [
            'k1' => [
                'min' => DataMustahik::min('k1'),
                'max' => DataMustahik::max('k1')
            ],
            'k2' => [
                'min' => DataMustahik::min('k2'),
                'max' => DataMustahik::max('k2')
            ],
            'k3' => [
                'min' => DataMustahik::min('k3'),
                'max' => DataMustahik::max('k3')
            ],
            'k4' => [
                'min' => DataMustahik::min('k4'),
                'max' => DataMustahik::max('k4')
            ],
            'k5' => [
                'min' => DataMustahik::min('k5'),
                'max' => DataMustahik::max('k5')
            ]
        ];

        return $minMaxValues;
    }
    private function normalize($data)
    {

        $minmax = $this->getMinMaxValues();

        $normal = [];
        foreach ($data as $key => $value) {
            $normal[$value->id] = [
              'k1' => ($value->k1 - $minmax['k1']['min']) / ($minmax['k1']['max']
- $minmax['k1']['min']),
              'k2' => ($value->k2 - $minmax['k2']['min']) / ($minmax['k2']['max']
- $minmax['k2']['min']),
              'k3' => ($value->k3 - $minmax['k3']['min']) / ($minmax['k3']['max']
- $minmax['k3']['min']),
```

```
            'k4' => ($value->k4 - $minmax['k4']['min']) / ($minmax['k4']['max']
- $minmax['k4']['min']),
            'k5' => ($value->k5 - $minmax['k5']['min']) / ($minmax['k5']['max']
- $minmax['k5']['min']),
          ];
      }
      return $normal;
```

Source Code Proses *Modified K-Nearest Neighbors*

```
private function euclideanDistance($point1, $point2)
    {
        return sqrt(
         pow($point1['k1'] - $point2['k1'], 2) +
        pow($point1['k2'] - $point2['k2'], 2) +
         pow($point1['k3'] - $point2['k3'], 2) +
         pow($point1['k4'] - $point2['k4'], 2) +
         pow($point1['k5'] - $point2['k5'], 2)
        );
    }
    private function perhitungan_mknn()
    {
        $data = DataMustahik::where('data_uji', 0)->get();
        $data_object_with_id = [];
        foreach ($data as $key => $value) {
            $data_object_with_id[$value->id] = $value;
        }
        //get max min to each k1
        //get normal value
        $normal = $this->normalize($data);

        $data_uji = DataMustahik::where('data_uji', 1)->get();
        $normal_uji = $this->normalize($data_uji);
        //cari bobot
        $bobot = [];
        foreach ($data as $point) {
            $bobot[$point->id] = $point->cluster == 'layak' ? 1 : 0;
        }
        $uji_distance = [];
        foreach ($normal_uji as $id_uji => $point_uji) {
        foreach ($normal as $id => $point) {
         $distance = $this->euclideanDistance($point_uji, $point);
         $hasil = [
          'distance' => $distance,
         'wv' => (1 / ($distance + 0.5)),//nilai validitas masukkan sini
            'id' => $id,
            'cluster' => $data_object_with_id[$id]->cluster,
            'mustahik' => $data_object_with_id[$id]->nama_mustahik
```

```
                ];
                $uji_distance[$id_uji][$id] = $hasil;
            // $distance = $this->euclideanDistance($point_uji, $point);
                // You can store or use the distance as needed
```