Classifier Implementation in PHP

<?php

declare(strict\_types=1);

namespace Phpml\Classification;

use Phpml\Helper\Predictable;

use Phpml\Helper\Trainable;

use Phpml\Math\Distance;

use Phpml\Math\Distance\Euclidean;

class KNearestNeighbors implements Classifier

{

use Trainable;

use Predictable;

/\*\*

\* @var int

\*/

private $k;

/\*\*

\* @var Distance

\*/

private $distanceMetric;

/\*\*

\* @param Distance|null $distanceMetric (if null then Euclidean distance as default)

\*/

public function \_\_construct(int $k = 3, ?Distance $distanceMetric = null)

{

if ($distanceMetric === null) {

$distanceMetric = new Euclidean();

}

$this->k = $k;

$this->samples = [];

$this->targets = [];

$this->distanceMetric = $distanceMetric;

}

/\*\*

\* @return mixed

\*/

protected function predictSample(array $sample)

{

$distances = $this->kNeighborsDistances($sample);

$predictions = (array) array\_combine(array\_values($this->targets), array\_fill(0, count($this->targets), 0));

foreach (array\_keys($distances) as $index) {

++$predictions[$this->targets[$index]];

}

arsort($predictions);

reset($predictions);

return key($predictions);

}

/\*\*

\* @throws \Phpml\Exception\InvalidArgumentException

\*/

private function kNeighborsDistances(array $sample): array

{

$distances = [];

foreach ($this->samples as $index => $neighbor) {

$distances[$index] = $this->distanceMetric->distance($sample, $neighbor);

}

asort($distances);

return array\_slice($distances, 0, $this->k, true);

}

}

<?php

require\_once \_\_DIR\_\_ . '/vendor/autoload.php';

use Phpml\Classification\KNearestNeighbors;

$q = $\_GET['q'];

$t = $\_GET['time'];

$samples = [[30, 0], [1, 10], [3, 8], [31, 3], [20, 1], [9, 8], [9, 9], [16, 0], [32, 4], [44, 0], [37, 2], [48, 2], [50, 4], [18, 0], [2, 7], [6, 0], [22, 6], [9, 3], [37, 2], [10, 6], [11, 6], [13, 6], [28, 2], [28, 8], [28, 3]];

$labels = ['T', 'T', 'T', 'T', 'NT', 'T', 'MT', 'MT', 'MT', 'T', 'MT', 'T', 'T', 'T', 'T', 'T', 'NT', 'MT', 'NT', 'T', 'MT', 'T', 'MT', 'T', 'T', 'NT', 'NT', 'T', 'T', 'NT', 'NT', 'T', 'T', 'T', 'T', 'T', 'T', 'T', 'MT', 'T', 'MT', 'T', 'T', 'MT', 'MT', 'MT', 'T', 'MT', 'T'];

//Training Data

$classifier = new KNearestNeighbors();

$classifier->train($samples, $labels);

//classifier

echo $classifier->predict([$t, $q])

?>