MACHINE LEARNING WITH PHP-ML

DERAK KILGO

https://www.flickr.com/photos/64064794@N05/26643212064

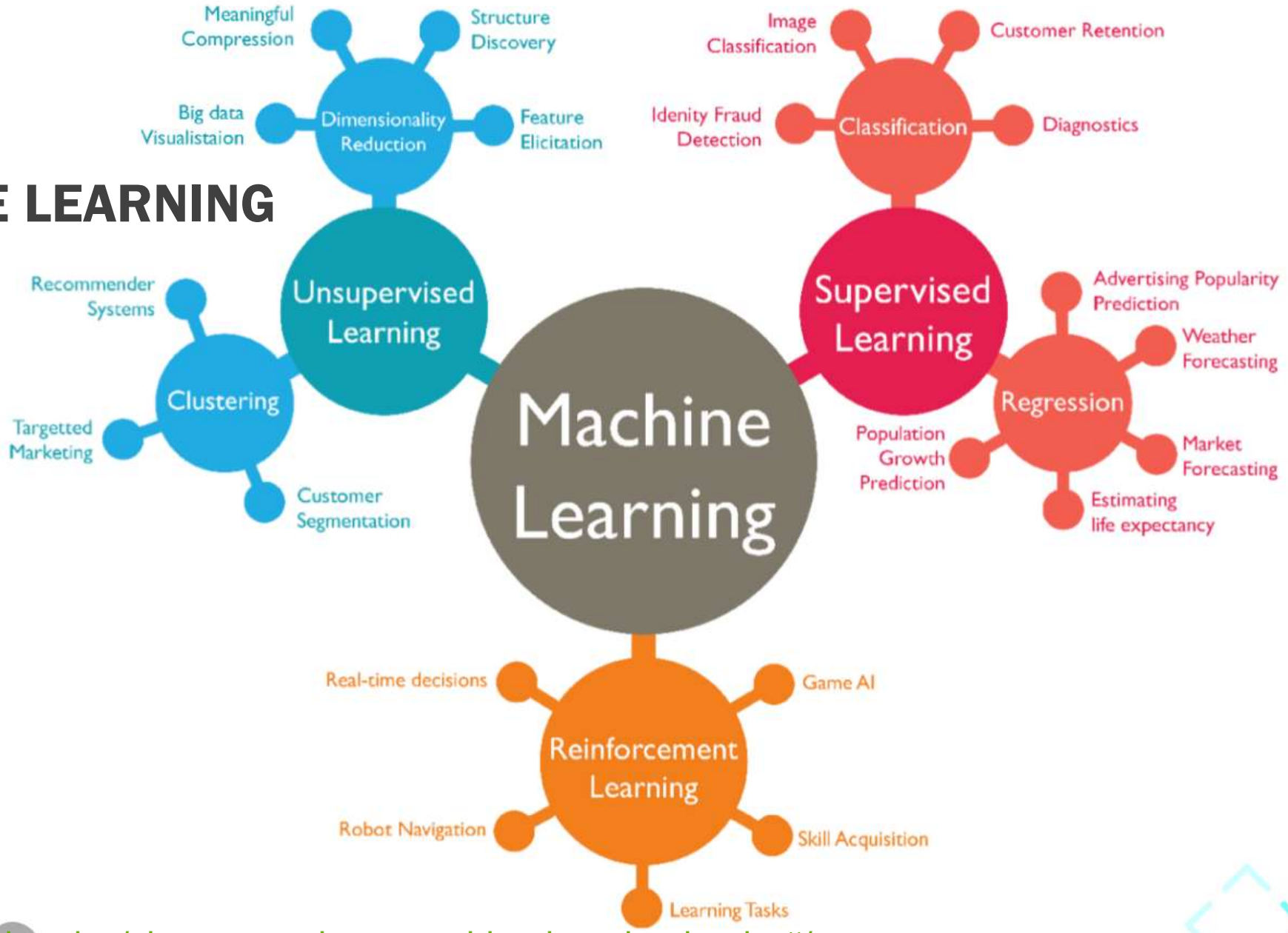
WHAT IS MACHINE LEARNING?

- Machine learning (ML) is a component of artificial intelligence (AI) that "learns" or adapts over time based on experience(inputs) instead of being explicitly programmed.
- It is widely used to offer custom experiences based on user interaction.
- Tech was born in the 1959 at IBM

WHY NOW?

- Learning is computationally expensive. Cloud computing commoditized CPU access.
- We have access to more data than ever before.
- Case in point: I “borrowed” 16 cores and 64GB of memory on amazon EC2 for few hours to test my sample models.
  (It wasn’t enough for my 3M row dataset.)

TYPES OF MACHINE LEARNING



Source: https://slides.com/arkadiuszkondas/shor-story-about-machine-learning-in-php#/

SUPERVISED LEARNING

- An algorithm builds a model from a set of data that contains both input and the desired outputs.
- Classification and Regression algorithms are types of supervised learning

UNSUPERVISED LEARNING

- An algorithm builds a model data that only contains the input. This type of model can find structure or patterns in the data like grouping or clustering.
- Clustering and Dimensionality Reduction are examples.

REINFORCEMENT LEARNING

- This is game ai and real-time decision making, and robot navigation.
- The algorithm will:
  - Observe.
  - Select actions using a policy.
  - Get a reward or penalty.
  - Update its policy (learning).
  - Iterate until an optimal policy is found.

COMMON ML TOOLKITS

- Amazon ML
- Amazon SageMaker
- Azure ML
- Google Prediction API
- SAS Enterprise Miner
- Splunk

- Scikit-Learn (python)
- Pandas (python)
- NumPy (python)
- Pytorch (python)
- Apache Spark MLlib (java)
- TensorFlow (c++/python)
- TensorFlow.js (Javascript)
- R (language)

PHP-ML : A MACHINE LEARNING LIBRARY FOR PHP

- Developed by “code craftsman” Arkadiusz Kondas in 2016
- https://php-ml.org/
- He calls it a “Fresh approach to Machine Learning in PHP. Algorithms, Cross Validation, Neural Network, Preprocessing, Feature Extraction and much more in one library.”
- Requires PHP 7.2 or greater.
- Includes data sets which can be used for learning how to use these tools.
- All of the code samples in this presentation are from the documentation.

ML TERMS

- **Samples** – Input to process (e.g text,images, sound video). Must have a fixed set of features.
- **Features** - Traits used to describe each item in a sample in a quantitative manner.
- **Feature Extraction** – Preparation of data; transforming and simplifying it.
- **Training** – The process of providing data to a ML algorithm for the purposes of “teaching” a desired outcome.

PROBLEMS SOLVED BY PHP-ML: REGRESSION

```
use Phpml\Regression\SVR;
use Phpml\SupportVectorMachine\Kernel;

$samples = [[60], [61], [62], [63], [65]];
$targets = [3.1, 3.6, 3.8, 4, 4.1];
$regression = new SVR(Kernel::LINEAR);
$regression->train($samples, $targets);
$regression->predict([64]) // return 4.03
```

PROBLEMS SOLVED BY PHP-ML: CLUSTERING

```
$samples = [[1, 1], [8, 7], [1, 2], [7, 8], [2, 1], [8, 9]];

//Or if you need to keep your identifiers along with yours samples you can use array keys as labels.

$samples = [ 'Label1' => [1, 1], 'Label2' => [8, 7], 'Label3' => [1, 2]];
$kmeans = new KMeans(2);
$kmeans->cluster($samples);
// return [0=>[[1, 1], ...], 1=>[[8, 7], ...]]
//or [0=>['Label1' => [1, 1], 'Label3' => [1, 2], ...], 1=>['Label2' => [8, 7], ...]]
```

PROBLEMS SOLVED BY PHP-ML: CLASSIFICATION

```
require_once __DIR__ . '/vendor/autoload.php';

use Phpml\Classification\KNearestNeighbors;

$samples = [[1, 3], [1, 4], [2, 4], [3, 1], [4, 1], [4, 2]];
$labels = ['a', 'a', 'a', 'b', 'b', 'b'];

$classifier = new KNearestNeighbors();
$classifier->train($samples, $labels);

echo $classifier->predict([3, 2]); // return 'b'
```

ML IS ALL ABOUT THE PROCESS

- Define the problem
- Gather data
- Prepare data
- Select an algorithm
- Train the model
- Validate the model

https://xkcd.com/1838/

MY PROBLEM: SENTIMENT ANALYSIS

- Determine if a review is positive or negative.
- We get survey data from every participant about their experience.
- We'd like to follow up with anyone having a negative experience.
- We host over 20K teachers at 600 events a year.

GATHER DATA

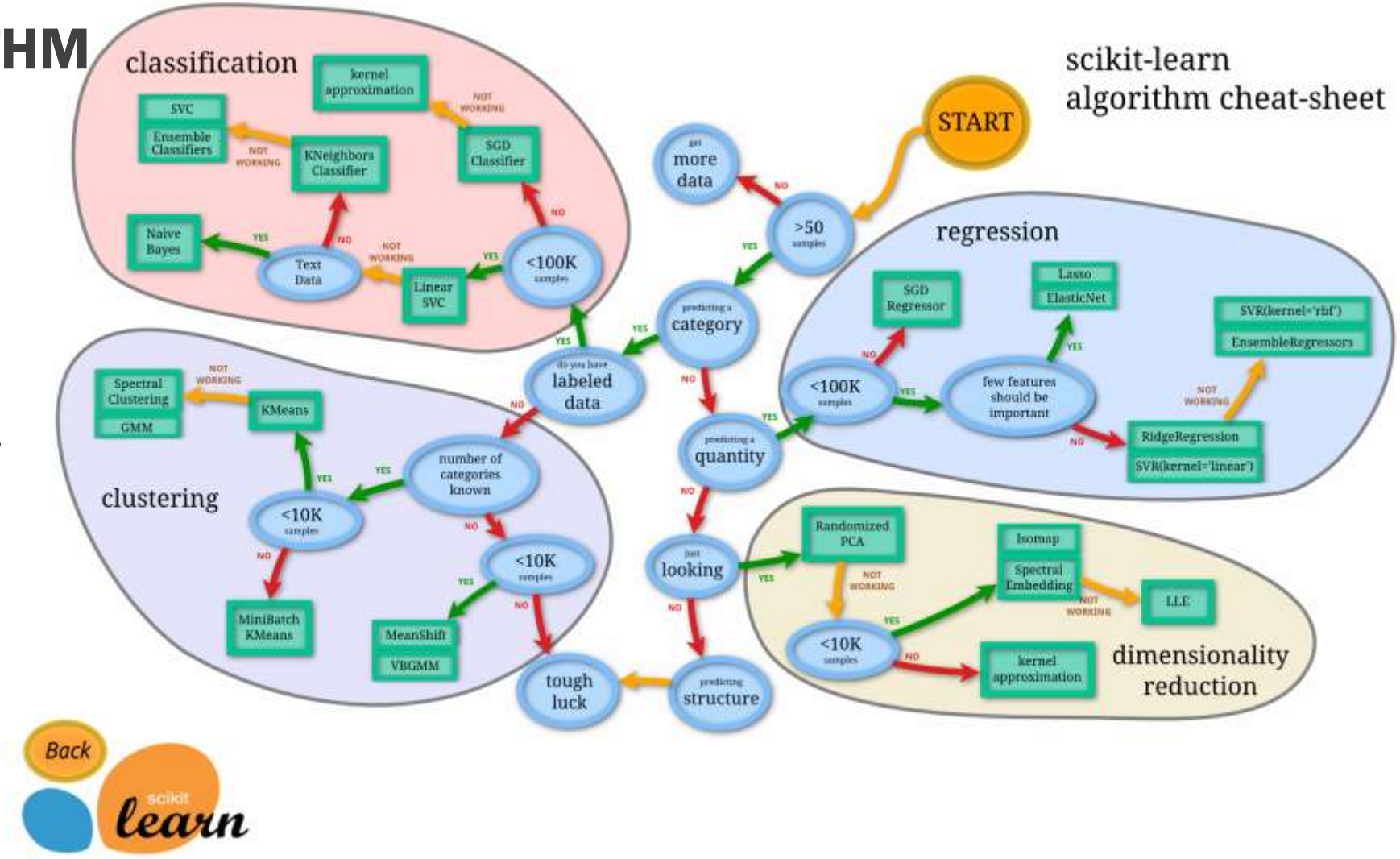
- Our data does not have labels.
- Kaggle.com offers training data which looks like our survey data. https://www.kaggle.com/bittlingmayer/amazonreviews
- This training set is 3.5M amazon reviews labeled positive/negative.

PREPARE THE DATA

- Format the data for input.
  Stripped double quotes and converted to CSV.
- Used “csvlint” to validate format. (https://github.com/Clever/csvlint)
- PHP-ML’s CSV dataset requires ‘feature’,’label’ format.
  This data is “label, feature”. Implement custom CSV import class to flip format.
- 3.5M rows is too big for my system.
  Use gnu “split” command to make smaller files out of this set.
  (1000 rows per file.)

SELECT AN ALGORITHM

- Naïve Bays is recommended for text processing



source: http://scikit-learn.org/stable/tutorial/machine_learning_map/index.html

TRAIN THE MODEL

```
echo "Loading data...\n";
$dataset = new CustomCsv(0,1,
    __DIR__ . '/data/amazonreviews/split/xaa',
    false, ',', 500);

$split = new StratifiedRandomSplit($dataset, 0.1);
unset($dataset);

echo “Assemble Pipeline...\n";
$pipeline = new Pipeline([
    new TokenCountVectorizer(new NGramTokenizer(3, 5), new English()),
    new TfIdfTransformer()
], new NaiveBayes());

echo “Training Model...\n";
$pipeline->train($split->getTrainSamples(),$split->getTrainLabels());

echo “Testing model ...\n";
$predicted = $pipeline->predict($split->getTestSamples());
echo 'Accuracy: ' . Accuracy::score($split->getTestLabels(), $predicted) . "\n";

echo "Saving model...\n";
$modelManager = new ModelManager();
$modelManager->saveToFile($pipeline, __DIR__ . '/data/amazonreview.phpml');
```

TRAIN THE MODEL

Load csv file into an array.
Only load the first 500 characters of each line.

```
echo "Loading data...\n";
$dataset = new CustomCsv(0,1,
    __DIR__ . '/data/amazonreviews/split/xaa',
    false, ',', 500);

$split = new StratifiedRandomSplit($dataset, 0.1);
unset($dataset);

echo “Assemble Pipeline...\n";
$pipeline = new Pipeline([
    new TokenCountVectorizer(new NGramTokenizer(3, 5), new English()),
    new TfIdfTransformer()
], new NaiveBayes());

echo “Training Model...\n";
$pipeline->train($split->getTrainSamples(),$split->getTrainLabels());

echo “Testing model ...\n";
$predicted = $pipeline->predict($split->getTestSamples());
echo 'Accuracy: ' . Accuracy::score($split->getTestLabels(), $predicted) . "\n";

echo "Saving model...\n";
$modelManager = new ModelManager();
$modelManager->saveToFile($pipeline, __DIR__ . '/data/amazonreview.phpml');
```

TRAIN THE MODEL

```
echo "Loading data...\n";
$dataset = new CustomCsv(0,1,
    __DIR__ . '/data/amazonreviews/split/xaa',
    false, ',', 500);

$split = new StratifiedRandomSplit($dataset, 0.1);
unset($dataset);

echo “Assemble Pipeline...\n";
$pipeline = new Pipeline([
    new TokenCountVectorizer(new NGramTokenizer(3, 5), new English()),
    new TfIdfTransformer()
], new NaiveBayes());

echo “Training Model...\n";
$pipeline->train($split->getTrainSamples(),$split->getTrainLabels());

echo “Testing model ...\n";
$predicted = $pipeline->predict($split->getTestSamples());
echo 'Accuracy: ' . Accuracy::score($split->getTestLabels(), $predicted) . "\n";

echo "Saving model...\n";
$modelManager = new ModelManager();
$modelManager->saveToFile($pipeline, __DIR__ . '/data/amazonreview.phpml');
```

Load csv file into an array.
Only load the first 500 characters of each line.

This will split our data 90/10 into train and test datasets. It will also choose a distribution of results for test that matches what’s in train. E.g: if 30% of the values in train or “positive”; 30% in test will also be positive.

TRAIN THE MODEL

```
echo "Loading data...\n";
$dataset = new CustomCsv(0,1,
    __DIR__ . '/data/amazonreviews/split/xaa',
    false, ',', 500);

$split = new StratifiedRandomSplit($dataset, 0.1);
unset($dataset);

echo “Assemble Pipeline...\n";
$pipeline = new Pipeline([
    new TokenCountVectorizer(new NGramTokenizer(3, 5), new English()),
    new TfIdfTransformer()
], new NaiveBayes());

echo “Training Model...\n";
$pipeline->train($split->getTrainSamples(),$split->getTrainLabels());

echo “Testing model ...\n";
$predicted = $pipeline->predict($split->getTestSamples());
echo 'Accuracy: ' . Accuracy::score($split->getTestLabels(), $predicted) . "\n";

echo "Saving model...\n";
$modelManager = new ModelManager();
$modelManager->saveToFile($pipeline, __DIR__ . '/data/amazonreview.phpml');
```

Load csv file into an array.
Only load the first 500 characters of each line.

This will split our data 90/10 into train and test datasets. It will also choose a distribution of results for test that matches what’s in train. E.g: if 30% of the values in train or “positive”; 30% in test will also be positive.

A pipeline allows a group of transformations to be applied automatically.

TRAIN THE MODEL

```
echo "Loading data...\n";
$dataset = new CustomCsv(0,1,
    __DIR__ . '/data/amazonreviews/split/xaa',
    false, ',', 500);

$split = new StratifiedRandomSplit($dataset, 0.1);
unset($dataset);

echo “Assemble Pipeline...\n";
$pipeline = new Pipeline([
    new TokenCountVectorizer(new NGramTokenizer(3, 5), new English()),
    new TfIdfTransformer()
], new NaiveBayes());

echo “Training Model...\n";
$pipeline->train($split->getTrainSamples(),$split->getTrainLabels());

echo “Testing model ...\n";
$predicted = $pipeline->predict($split->getTestSamples());
echo 'Accuracy: ' . Accuracy::score($split->getTestLabels(), $predicted) . "\n";

echo "Saving model...\n";
$modelManager = new ModelManager();
$modelManager->saveToFile($pipeline, __DIR__ . '/data/amazonreview.phpml');
```

Load csv file into an array.
Only load the first 500 characters of each line.

This will split our data 90/10 into train and test datasets. It will also choose a distribution of results for test that matches what’s in train. E.g: if 30% of the values in train or “positive”; 30% in test will also be positive.

A pipeline allows a group of transformations to be applied automatically.

Use our “train” samples to train the “Naïve Bayes” model.

VALIDATE THE MODEL

```
echo "Loading data...\n";
$dataset = new CustomCsv(0,1,
    __DIR__ . '/data/amazonreviews/split/xaa',
    false, ',', 500);

$split = new StratifiedRandomSplit($dataset, 0.1);
unset($dataset);

echo “Assemble Pipeline...\n";
$pipeline = new Pipeline([
    new TokenCountVectorizer(new NGramTokenizer(3, 5), new English()),
    new TfIdfTransformer()
], new NaiveBayes());

echo “Training Model...\n";
$pipeline->train($split->getTrainSamples(),$split->getTrainLabels());

echo “Testing model ...\n";
$predicted = $pipeline->predict($split->getTestSamples());
echo 'Accuracy: ' . Accuracy::score($split->getTestLabels(), $predicted) . "\n";

echo "Saving model...\n";
$modelManager = new ModelManager();
$modelManager->saveToFile($pipeline, __DIR__ . '/data/amazonreview.phpml');
```

Load csv file into an array.
Only load the first 500 characters of each line.

This will split our data 90/10 into train and test datasets. It will also choose a distribution of results for test that matches what’s in train. E.g: if 30% of the values in train or “positive”; 30% in test will also be positive.

A pipeline allows a group of transformations to be applied automatically.

Use our “train” samples to train the “Naïve Bayes” model.

These functional tests are crucial to understanding how accurate your model is.

SAVE THE MODEL

```
echo "Loading data...\n";
$dataset = new CustomCsv(0,1,
    __DIR__ . '/data/amazonreviews/split/xaa',
    false, ',', 500);

$split = new StratifiedRandomSplit($dataset, 0.1);
unset($dataset);

echo “Assemble Pipeline...\n";
$pipeline = new Pipeline([
    new TokenCountVectorizer(new NGramTokenizer(3, 5), new English()),
    new TfIdfTransformer()
], new NaiveBayes());

echo “Training Model...\n";
$pipeline->train($split->getTrainSamples(),$split->getTrainLabels());

echo “Testing model ...\n";
$predicted = $pipeline->predict($split->getTestSamples());
echo 'Accuracy: ' . Accuracy::score($split->getTestLabels(), $predicted) . "\n";

echo "Saving model...\n";
$modelManager = new ModelManager();
$modelManager->saveToFile($pipeline, __DIR__ . '/data/amazonreview.phpml');
```

Load csv file into an array.
Only load the first 500 characters of each line.

This will split our data 90/10 into train and test datasets. It will also choose a distribution of results for test that matches what’s in train. E.g: if 30% of the values in train or “positive”; 30% in test will also be positive.

A pipeline allows a group of transformations to be applied automatically.

Use our “train” samples to train the “Naïve Bayes” model.

These functional tests are crucial to understanding how accurate your model is.

PHP-ML supports “saving” a model if you wish to skip the training step next time. These files can be large.

CLOSER LOOK AT THE PIPELINE

```
$pipeline = new Pipeline([
    new TokenCountVectorizer(new NGramTokenizer(3, 5), new English()),
    new TfIdfTransformer()
], new NaiveBayes());
```

TokenCountVectorizer is a method of “feature extraction”.
It will convert our text into a vocabulary and array of token counts.

```
$samples = [
    'Lorem ipsum dolor sit amet dolor',
    'Mauris placerat ipsum dolor',
    'Mauris diam eros fringilla diam',
];

$vectorizer = new TokenCountVectorizer(new WhitespaceTokenizer());

// Build the dictionary.
$vectorizer->fit($samples);

// Transform the provided text samples into a vectorized list.
$vectorizer->transform($samples);
// return $samples = [
//    [0 => 1, 1 => 1, 2 => 2, 3 => 1, 4 => 1],
//    [5 => 1, 6 => 1, 1 => 1, 2 => 1],
//    [5 => 1, 7 => 2, 8 => 1, 9 => 1],
//];
```

https://github.com/php-ai/php-ml/blob/master/docs/machine-learning/feature-extraction/token-count-vectorizer.md

CLOSER LOOK AT THE PIPELINE

```
$pipeline = new Pipeline([
    new TokenCountVectorizer(new NGramTokenizer(3, 5), new English()),
    new TfIdfTransformer()
], new NaiveBayes());
```

```
$stopWords = [
        'a', 'about', 'above', 'after', 'again', 'against', 'all', 'am', 'an', 'and', 'any', 'are',
'aren\'t', 'as', 'at', 'be', 'because',
        'been', 'before', 'being', 'below', 'between', 'both', 'but', 'by', 'can\'t', 'cannot',
'could', 'couldn\'t', 'did', 'didn\'t',
        'do', 'does', 'doesn\'t', 'doing', 'don\'t', 'down', 'during', 'each', 'few', 'for',
'from', 'further', 'had', 'hadn\'t', 'has',
        'hasn\'t', 'have', 'haven\'t', 'having', 'he', 'he\'d', 'he\'ll', 'he\'s', 'her', 'here',
'here\'s', 'hers', 'herself', 'him',
        'himself', 'his', 'how', 'how\'s', 'i', 'i\'d', 'i\'ll', 'i\'m', 'i\'ve', 'if', 'in', 'into', 'is',
'isn\'t', 'it', 'it\'s', 'its',
        'itself', 'let\'s', 'me', 'more', 'most',.....
```

English() provides a list of “stop words” or words with little contextual significance. We remove these words to improve our model.

https://github.com/php-ai/php-ml/blob/master/src/FeatureExtraction/StopWords/English.php

https://github.com/php-ai/php-ml/blob/master/docs/machine-learning/feature-extraction/token-count-vectorizer.md

CLOSER LOOK AT THE PIPELINE

```
$pipeline = new Pipeline([
    new TokenCountVectorizer(new NGramTokenizer(3, 5), new English()),
    new TfIdfTransformer()
], new NaiveBayes());
```

```
use Phpml\FeatureExtraction\TfIdfTransformer;

$samples = [[1, 2, 4],[0, 2, 1]];

$transformer = new TfIdfTransformer($samples);
$transformer->transform($samples);

/*
$samples = array (
  0 =>  array (
    0 => 0.3010299956639812, 1 => 0.0, 2 => 0.0,
  ),
  1 =>  array (
    0 => 0.0, 1 => 0.0,  2 => 0.0,
  ),
)
  }
} */
```

TfIdfTransformer() is a method of "feature extraction". It's step 2 in our pipeline because it weights words that are more frequent; useful in a classification model.

Tf–idf, short for term frequency–inverse document frequency, is a numerical statistic that is intended to reflect how important a word is to a document in a collection or corpus.

The weight of a term that occurs in a document is simply proportional to the term frequency.

- https://en.wikipedia.org/wiki/Tf-idf
- https://php-ml.readthedocs.io/en/latest/machine-learning/feature-extraction/tf-idf-transformer/

https://github.com/php-ai/php-ml/blob/master/docs/machine-learning/feature-extraction/token-count-vectorizer.md

CLOSER LOOK AT THE PIPELINE

```
$pipeline = new Pipeline([
    new TokenCountVectorizer(new NGramTokenizer(3, 5), new English()),
    new TfIdfTransformer()
], new NaiveBayes());
```

```
$samples = [[5, 1, 1], [1, 5, 1], [1, 1, 5]];
$labels = ['a', 'b', 'c'];
$classifier = new NaiveBayes();
$classifier->train($samples, $labels);

$classifier->predict([3, 1, 1]);
// return 'a' – matched two factors in a single sample.

$classifier->predict([[3, 1, 1], [1, 4, 1]);
// return ['a', 'b'] – matched two factors in two samples.
```

NaiveBays() is our classifier algorithm

For example, a fruit may be considered to be an apple if it is red, round, and about 10 cm in diameter. A naive Bayes classifier considers each of these features to contribute independently to the probability that this fruit is an apple, regardless of any possible correlations between the color, roundness, and diameter features.

(Source Wikipedia)

https://php-ml.readthedocs.io/en/latest/machine-learning/classification/naive-bayes/

TEST IT

```
[ec2-user@ip-172-16-0-73 phpml]$ php ./test-phpml-amazon.php
Loading data...
assemble pipeline...
training model...

mmap() failed: [12] Cannot allocate memory

mmap() failed: [12] Cannot allocate memory
PHP Fatal error:  Out of memory (allocated 97148751872) (tried to allocate 75497472 bytes) in /home/ec2-user/phpml/
raction/TokenCountVectorizer.php on line 92
[ec2-user@ip-172-16-0-73 phpml]$ cat /proc/meminfo
MemTotal:       65326868 kB
MemFree:        65062656 kB
MemAvailable:   64638484 kB
Buffers:            2592 kB
Cached:            43664 kB
SwapCached:            0 kB
```

Because these models are 100% in memory; they can use up a lot of space quickly.

PHP-ML AS A SERVICE WITH REACTPHP

- Share-nothing doesn’t work well for the very large modal object.
- It would be more efficient to keep a modal in memory and query it.
- ReactPHP allows us to bring up a model and use it to answer many questions as a web service.

OTHER METHODS OF SENTIMENT ANALYSIS

- There are other statistical, non-ML ways to determine sentiment with php.
- AFINN-based sentiment analysis. (certifiedwebninja/caroline)
- Lexicon and rule-based sentiment analysis tool using VADER (Valence Aware Dictionary and sentiment Reasoner) (davmixcool/php-sentiment-analyzer)
- Dictionary based. (risan/sentiment-analysis)

OTHER METHODS OF SENTIMENT ANALYSIS

- Using the testing tools from PHP-ML; we can test each of these methods by constructing a wrapper class that implements the “Classifier” interface.
- Because these methods rely on pre-built word lists; we need to remove our transformations from the pipeline before testing.
- Results were worse than our PHP-ML model with 65% accuracy across all the libraries.

THANK YOU

- Code and slides can be found on GitHub
  https://github.com/derak-kilgo/phpml-sentiment-demo
- Developers blog has lots of code samples and slides from previous talks.
  https://arkadiuszkondas.com
- PHP-ML on GitHub
  https://github.com/php-ai/php-ml
- PHP-ML Docs
  https://php-ml.readthedocs.io/en/latest/
- https://www.slideshare.net/Simplilearn/what-is-machine-learning-machine-learning-basics-machine-learning-algorithms-simplilearn