## 2 Coursework Documentation – Artificial Intelligence

#### Middlesex University – London

#### Department of Computer Science

#### School of Science and Technology

#### Author

### Adam Jarzebak

## Email

### [aj695@live.mdx.ac.uk](mailto:aj695@live.mdx.ac.uk)

## Student Number

### M00525614

## Date of submission

### 9 March 2018 – Friday

# CSD 3939 Course Work 2

## Machine Learning

In this coursework I was attempting to solve problem of recognition for handwritten digits. It is widely known problem which is continually in progress and research area. Many researchers are trying to improve an algorithm and enhance accuracy for a recognition.

Most of them listed on the website: <http://yann.lecun.com/exdb/mnist/> .There are also included all results.

My solution if different from other. Only 2 days before deadline I changed my direction and from support vector machine started on improving Euclidean distance algorithm. After many attempts and implementations of different algorithms I decided to choose Euclidean distance algorithm and plug to this cosine similarity algorithm. Which turned out to be very successful step. For entire data set, which was provide for this coursework I achieved 99.82% accuracy.

Please find more detailed results in Figure1.

Much the same as in programming we initially figure out how to print "hi, world!", likewise in machine learning we initially do handwritten digit recognition (MNIST). In this report I will utilize one of the least difficult machine learning calculations called k closest neighbours to take care of this popular issue of perceiving handwritten digits.

It is an order calculation which means that it groups the new information point (test data) into some class. To do as such it essentially takes a gander at its (test data) separate from other data focuses (preparing focuses). At that point out of the k nearest preparing focuses the class in lion's share is allocated to that new test information point.

The separation metric relies upon the issue decided for grouping errand. This is on the grounds that, for example, the separation between the aptitudes of two software engineers (scores in rivalry) would be not quite the same as the separation between two kites flying in sky (3D dimensions), and so forth. Similarly, the distance metric for distance between images is also different (as discussed below).

That is about k closest neighbours! The calculation is entirely straightforward.

The distance between the two images can be calculated in many different ways. But I used the Euclidean distance algorithm. The way it will be calculated for images is by summing the squared Euclidean distance between the corresponding pixels of two images(digits).

If I find the right digit then I move on, to the next data, however If I do not, then I run the same loop but with another metric algorithm which is called cosine similarity algorithm.

And eventually I get more accurate results.

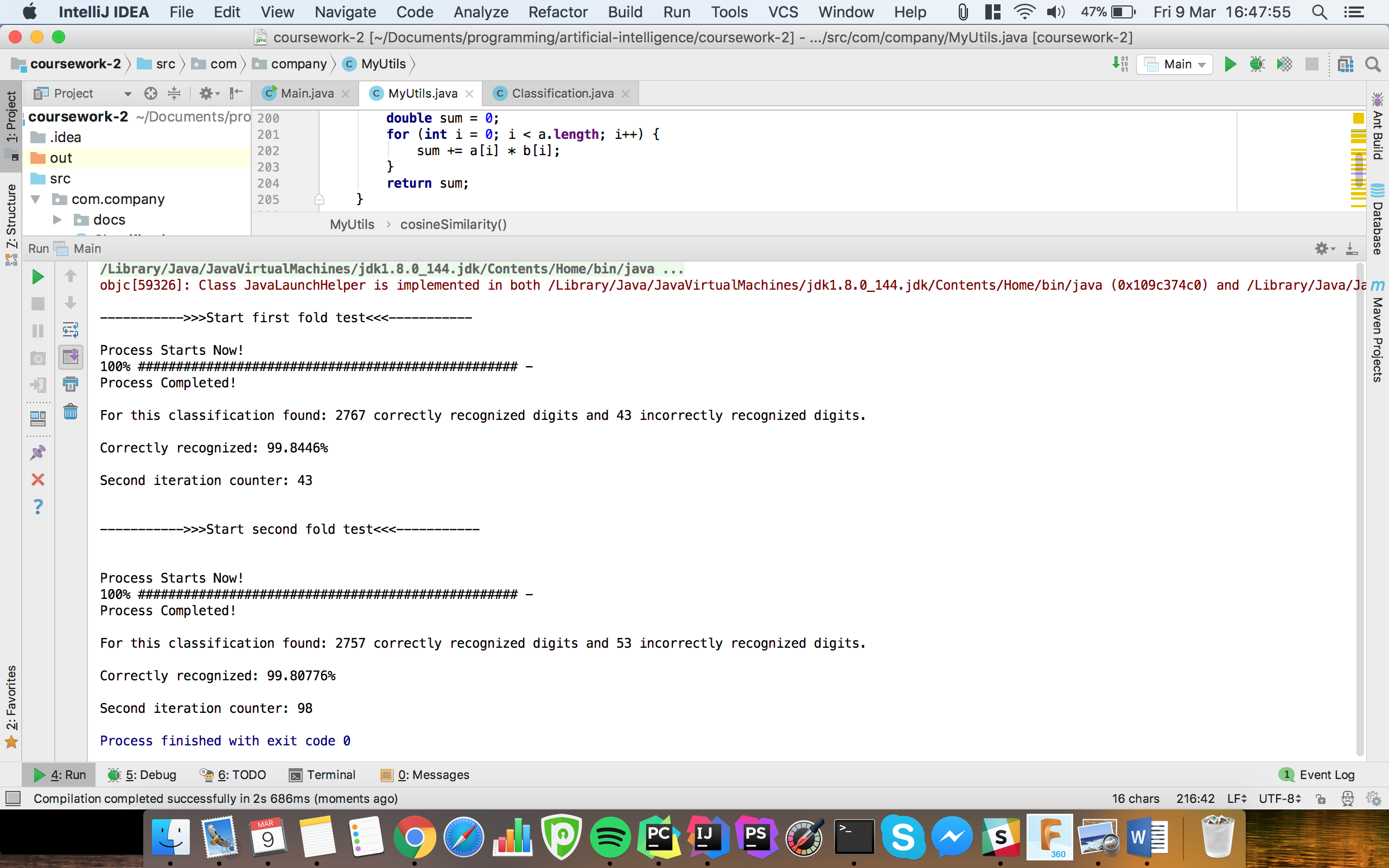


Figure 1Results

Also, as a part of my coursework, I implemented web services where the algorithm can be tested on real digits. This can be found on: ai.jarzebak.eu

Entire documentation of the coding is included on web service as well and can be found on jarzebak.eu/aidocs

Lastly, I tried implement SVM algorithm however due to bug I could not move on and is only working partially. All code is included so it can be reviewed.