

Assignment 2 – CSC3060 “AIDA”

Worth 30% of the module assessment. Assignment is marked out of 100 marks.

Deadline: 11pm Friday, 15th November 2019.

This version: 2019-10-03

Introduction

In this assignment, you will:

- (a) Create a dataset of doodles (which you will use for your analyses and experiments in the rest of Assignment 2, and in Assignment 3).
- (b) Calculate features (i.e. variables) from the doodles which may be useful for identifying the objects depicted in doodles automatically.
- (c) Perform statistical analysis of the datasets, using methods of statistical inference.

When you use a procedure that has an element of randomness, please use the seed value 3060 (your code should give the same results each time it runs).

Sections 1 and 2 of this Assignment can be completed in one of the following programming languages: **Python, R, Java**. Section 3 must be completed in **R**.

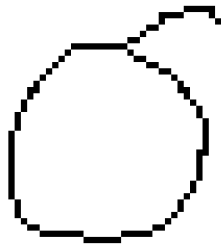
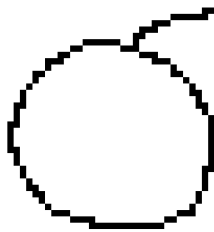
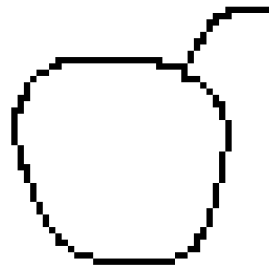
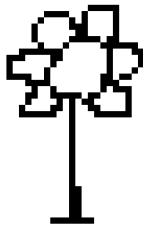
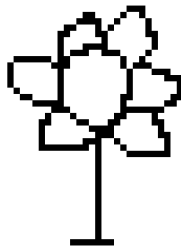
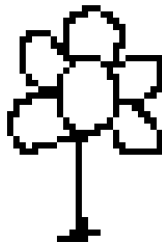
Please read carefully the information about the assessment criteria and marking process at the end of this document.

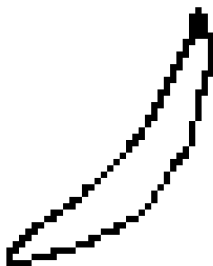
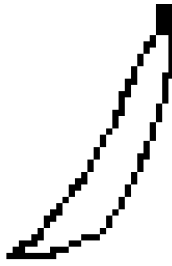

Section 1 (10 marks): Creating a dataset

This section asks you to build a dataset of images composed of doodles of common everyday objects.

The goal is to create a dataset containing images for each of 4 **living thing** objects {cherry, pear, banana, flower} and 4 **nonliving thing** objects {envelope, golfclub, pencil, wineglass}. You will create 20 example images for each of the 8 classes, giving a total of 160 images. Each image should be obtained by hand-drawing the image yourself (preferably with a touch screen, using the lab computers, although it is fine if you create them using the computer mouse). The quality of the drawing is not essential, as long as the objects can easily be easily identified. The images for each of the eight objects should follow the same basic canonical form as the examples presented on the next page, and should be in a natural orientation (as shown in the examples). The images will vary from sample to sample; however, each object should fit reasonably well in the 50x50 box (i.e. do not draw a tiny object in one corner of the 50x50 box; that would make your life easier when it comes to doing analyses!).

Each image is represented by a black & white matrix with size 50 rows by 50 columns. In the matrix, the number “1” represents black pixels and “0” represents white pixels. As such, one image can be stored in a plaintext “.csv” file containing the matrix (and no headers).

Class	<i>cherry</i>		
Example Images			
Class	<i>flower</i>		
Example Images			

Class	<i>banana</i>		
Example Images			



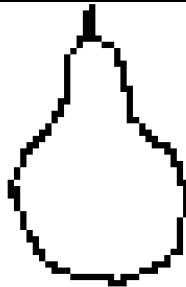



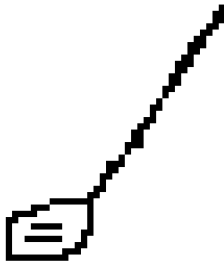
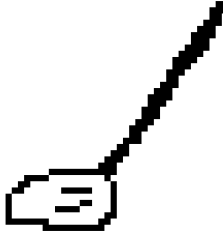



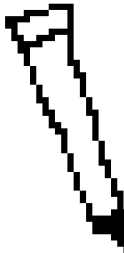
Class	<i>pear</i>		
Example Images			

Figure 1: Examples of doodles for the four living thing objects.

Class	<i>envelope</i>		
Example Images			

Class	<i>golfclub</i>		
Example Images			

Class	<i>pencil</i>		
Example Images			


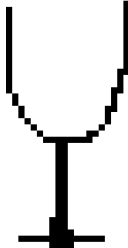

Class	<i>wineglass</i>		
Example Images			

Figure 2: Examples of doodles for the four non-living thing objects.

Each image should be stored in a tab-delimited csv file consisting of 50 rows, with 50 elements (i.e. 1 or 0) in each row, as in the example below.



Figure 3: A correctly formatted .csv file for a “cherry” image, displayed in Notepad++ (yellow arrows indicate tab characters).

You may use whatever means you prefer to obtain the images and .csv files. However, a suggestion is to use the software GIMP (<http://www.gimp.org>). GIMP is installed on the lab machines. Using GIMP, you can create a new image with 50 by 50 points (pt), advanced options 1 pixel/pt, color space grayscale, fill with background colour. This will give you a small white square, which you can magnify to e.g. 1000% in order to make it easier to draw on. To draw on the image, you can select the pencil tool and adjust the brush size to (e.g.) 1 pixel. The standard file formats of GIMP are useful to save the images, but we need a more easily readable format. One good option is to export as PGM, type ASCII. In this format, each image becomes a text file with a header consisting of the following four lines:

```
P2
# CREATOR: ...
50 50
255
```

The third and fourth lines of the header specify the pixel array size and the maximum allowed pixel value, respectively. (The images are grayscale, with 0 representing fully black and 255 representing fully white).¹

The remaining lines of the file specify the pixel values, with one value on each line; the total number of pixel values should correspond to the specified array size (i.e. 50*50=2500).

For our purposes, a number < 128 represents a black pixel, while a number ≥ 128 represents a white one. Such a format can be easily converted into a matrix containing ones and zeros, as presented in Figure 3 above. You shall save each image matrix as a csv file following the specification above, and using the filename STUDENTNR_LABEL_INDEX.csv, where STUDENTNR is your student number (e.g. 123456), INDEX is a two numeral code from ‘01’ to ‘20’, indexing the set of 20 images you must create for each object, and LABEL is the name of the object (i.e. one of the eight labels given in the examples

¹ For further information about this image format, see https://en.wikipedia.org/wiki/Netpbm_format

above, e.g. “golfclub”).

For example, if your student number is 123456, then **123456_golfclub_08.csv** would be the eighth image you created for the category ‘golfclub’. (As well as creating the csv files, you may also want to keep the PGM files, in case you need to inspect the data later on).

As part of your submission, upload the csv files that you create in a directory called “section1_images”, along with any code you wrote to create the csv files, in a folder called “section1_code” (see submission instructions at the end of this document).

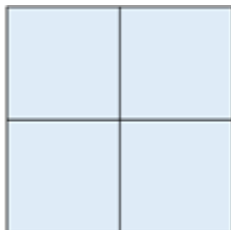
It is important to upload the images in the correct csv format as these files will be used to verify your calculations in the next section. File Encoding should be UTF8 (not UTF8-BOM or anything else). You can check the encoding in Notepad++.

In your report, very briefly (2-3 sentences) explain in your own words how you created the images and obtained the matrices from them.

Section 2 (40 marks): Feature engineering

Using each 50x50 matrix obtained from an image as described above, you must create an array of characteristics that describe some features of the image. Each feature will be a number (i.e. each feature is a numeric variable). There are 20 features in total.



Before we describe the features, let us define some vocabulary. An ***n-tile*** is an $n \times n$ selection from the image array. This is an example of a 2-tile:



Furthermore, each pixel has 8 ***neighbours***, which can be referred to as follows:

upper-left	upper	upper-right
left	[pixel]	right
lower-left	lower	lower-right

Features to be calculated (corresponding to columns of your features output file):

Feature Index	Feature Short Name	Feature Description
	label	The true name of the object in the image (represented by one of the 8 object names above). The label is not a true feature, and should not be used as a feature for statistical tests or during model training.
	index	The index of this image instance (a number from 1 to 20). The index is not a true feature, and should not be used as a feature for statistical tests or during model training.
1	nr_pix	The number of black pixels in the image.
2	height	The vertical distance between the topmost and bottommost black pixels in the image.
3	width	The horizontal distance between the leftmost and rightmost black pixels in the image.
4	span	The maximum Euclidean distance between any two black pixels in the image.
5	rows_with_5	Number of rows with five or more black pixels
6	cols_with_5	Number of columns with five or more black pixels
7	neigh1	Number of black pixels with exactly 1 neighbouring pixel
8	neigh5	Number of black pixels with 5 or more neighbours
9	left2tile	The number of unique 2-tiles in the image where the leftmost two entries are black and the rightmost two entries are white:  . Tiles may overlap.
10	right2tile	The number of unique 2-tiles in the image where the rightmost two entries are black and the leftmost two entries are white:  .
11	verticalness	The sum of the previous two features, divided by the number of black pixels in the image.
12	top2tile	The number of unique 2-tiles in the image where the top two entries are black and the bottom two entries are white.
13	bottom2tile	The number of unique 2-tiles in the image where the bottom two entries are black and the top two entries are white.
14	horizontalness	The sum of the previous two features, divided by the number of black pixels in the image.
15	[your label]	Define a custom feature using 3-tiles. Explain and justify the rationale for your feature.
16	[your label]	Define another custom feature using 3-tiles. Explain and justify the rationale for your feature.
17	nr_regions	Two black pixels A and B are connected if they are neighbours of each other, or if a black pixel neighbour of A is connected to B (this definition is actually symmetric); a connected region is a maximal set of black pixels which are connected to each other; this feature is the number of connected regions in the image.
18	nr_eyes	A region of white pixels is an eye if there is a ring of black pixels surrounding it which are all connected (i.e. they form a chain of neighbours). This feature is the number of eyes in the image. Note that while two neighbouring black pixels are necessarily in the same region, two neighbouring white pixels are not necessarily in the same eye.
19	hollowness	This is the number of white pixels which are in eyes divided by the number of black pixels in the image.
20	[your label]	Design any other feature you like. Justify why you think it may be a useful feature for distinguishing between object doodles.

Your task in this section is to write code to calculate each of the features above. In calculating pixel neighbours, you can assume that the images are padded on each side with white pixels.

Save your calculated features in a file called **STUDENTNR_features.csv**, where STUDENTNR is your student number. This file will consist of 161 rows. The first row gives the column names (i.e. the

strings in the **Feature Short Name** column in the table above). The remaining 160 rows list the tab-separated feature values for each of your 160 images. The first entry in the row will be the LABEL word, the second will be the image INDEX, and the remaining 20 entries will be the calculated features.

For example, the features for your eighth “cherry” image may be as follows:

```
cherry 8 34 12 14 12 1.1667 8 8 1 2 4 11 8 7 12 11 1 2 1 0.11 22
```

The 20 rows that correspond to the 20 instances of a particular object should be grouped together in the features file, and the order of the 20 rows should correspond to the INDEX used in the image filenames. In other words, the 160 rows of **STUDENTNR_features.csv** should be sorted first by the label (alphabetical order) and secondly by the index.

If you cannot calculate a particular feature, you may use a random integer between 0 and 10 for the feature values instead. (You will lose marks for not calculating the feature, but you can use the random values in the analyses that follow in the subsequent section. You should report that you have done this in the assignment report).

In your report, briefly describe and explain the code you have written to calculate the features above. If you ran into difficulties, you should still explain your thought processes and attempts to calculate the features. In the case of the custom features, you should explain your rationale for choosing the features you did, as well as how they are calculated (i.e. you should give a justification for why you think these features should be useful).

You should put the file **STUDENTNR_features.csv** in a folder called section2_features. Put code for this section in a folder called section2_code. Your code should use relative paths; i.e. it should read the image matrixes from “../section1_images” and save the feature file to “../section2_features”.

Section 3: Statistical analyses of feature data (50 marks)

In this section, you will perform statistical analyses of the feature data, in order to explore which features are important for distinguishing between different kinds of doodles.

You shall use descriptive statistics (mean, variance, etc.), null hypothesis testing, and confidence intervals to perform your analysis of the data. You are encouraged to provide tables, figures, and/or graphs in the report to support your discussions and findings. When performing tests, always consider whether multiple test correction is needed.

It is your responsibility to define the appropriate assumptions to run the tests, and to choose an appropriate test according to the data characteristics and the question that you are studying. In general, you will not be told what tests or R functions to use; it is up to you to explain and justify your choice. You are not necessarily restricted to the hypothesis tests that were discussed in the lectures. You may assume a significance level of 0.05 for the analyses when running hypothesis testing.

In particular, in the report you should address each of the following subtasks, using appropriate statistical tests, tables, graphs, etc.

1. Construct suitable histograms for the **nr_pix**, **height**, and **cols_with_5** features, for each of the following sets of items: (a) the full set of 80 living things, (b) the full set of 80 nonliving things (c) the entire set of 160 items. Briefly describe the shape of the distributions and comment on any interesting patterns across the datasets. Visually assess the skew and normality of the distributions.

2. Present useful summary statistics (e.g. mean and standard deviation) about all the features, for (a) the full set of 80 living things, (b) the full set of 80 nonliving things (c) the entire set of 160 items. Briefly discuss the summary statistics, and whether they already suggest which features may be useful for discriminating living and non-living images. For features you feel may be interesting for discrimination between groups, consider suitable visualisations (e.g. histogram of feature values for the groups²). State what type of variable (continuous, categorical, etc) each variable is.
3. Assume that the **nr_pix** variable is sampled from a population which is normally distributed. Estimate the mean and variance of the distribution from the available data for the 160 items. Plot the theoretical normal distribution for **nr_pix** using this mean and variance and compare to the corresponding histogram.
4. Assuming that the **nr_pix** variable is from a normally distributed population as above, what is the cut-off value for the **nr_pix** variable such that a randomly sampled image has a 5% probability of having a **nr_pix** value that is above that cut-off value?
5. Certain statistical tests assume that data are normally distributed. For each of the feature variables 1-14, identify variables with extreme skew and investigate transformations of the variables. Which features would you choose to transform? Explain how you reach your decision and how the transformation changes the distribution.
6. Investigate the relationship between the “height” variable and the “span” variable. Are these variables linearly associated? Consider suitable visualisation. Describe and conduct a suitable statistical test to measure the degree of association between these two variables.
7. Is the **nr_pix** feature useful to discriminate between the 4 different classes of *wineglass*, *golfclub*, *pencil*, and *envelope*? (Note that here we are looking for differences between 4 different groups, so consider a statistical method that tests for statistically significant differences between more than 2 groups). State clearly the statistical test used, the assumptions of the test, and how the assumptions relate to your data. If you use ANOVA, provide the full results table for the statistical test, and describe how each element of the results table is calculated.
8. Repeat (9) the statistical analysis reported in (9) above, but using a suitable randomization test, and using the “hollowness” feature (if you could not calculate hollowness in Section 1, you can use some other feature here). (You will use randomization to model the distribution of the F-statistic under the assumption of the null hypothesis, and compare the true F-statistic to this).
9. Are there features which are useful to discriminate between the set of living and nonliving things? (Consider statistical tests which test for differences between two groups). Briefly interpret your findings.
10. Fit a multiple regression model to predict **nr_pix** as best you can from some subset of the remaining variables. Briefly discuss your choice of variables.

For all questions above, you shall explain your reasoning, assumptions and steps of the procedure (including the statistical analysis) when preparing the report. Use statistics to justify your reasoning. If you are generating p-values for analysing the statistical significance of some features, make sure to explain how they were obtained. It is your task to decide and justify what the most appropriate inference to be performed in each case is, and to discuss the results you obtained.

Put code for this section in a folder called `section3_code`. Your code should use relative paths; i.e. it should read the feature data from `../section2_features`. It should be straightforward for the assessor to rerun your code to produce the same results as presented in your report. Ensure that the different subsections (1 to 20) are clearly labelled in the source code, and in the report. Each of these subtasks should be addressed in a separate subsection of your report.

² A nice example: <https://stackoverflow.com/questions/36049729/r-ggplot2-get-histogram-of-difference-between-two-groups>

Assessment criteria and marking process

The most important criteria in marking is the completeness, accuracy, quality and clarity of your report (approximately 75% weighting, across the full submission). In your report, you should clearly demonstrate that you understand the methods used in each sub-task. Explain your chosen approach, your reasoning, and the assumptions and steps of the procedures used. You should explain and interpret your results, demonstrating understanding and independent thinking. What are your results telling you? Are the results what you would expect? If you ran into difficulties, explain what they were and the efforts you made to try to overcome them.

Code has a weighting in marking of approximately 15%. Your code should be clear and logically organised, and do what is required, but code efficiency and code sophistication is not important (this assignment does not require complex programming). However, you should use loops and variables (rather than hard-coded values) where appropriate. If you use freely licenced code, packages, or libraries (which is encouraged), these should be appropriately referenced (e.g. by citing a URL in a comment). For example, using StackOverflow code snippets is fine, provide you acknowledge the use and provide the URL to the code snippets in the comments, and follow the MIT licence. The code must be easy to use and the comments must include information about the required steps to replicate the results that you have obtained and are presenting in your report (transparency and replicability are essential in data analysis). You may break the code in each folder into multiple source files if it is logical to do so, but no folder should have more than 5 source files.

Attention to detail and following the assignment instructions accurately will also be considered in marking (approximately 10% weighting). Each sub-task has a precise specification. Make sure you carefully follow the instructions, and use the features specified for each task, the specified procedures (seed value, data file specifications and file names, directory structure and names, etc). Make sure you upload your deliverable files in the specified formats.

Your report should explain how you have performed the analysis, but do not explain details of code implementation in your report - use the source code comments for that. Properly cite any sources that you have used.

Deliverables

You must submit your assignment online, using the module webpage, by 11pm Friday, 15th November 2019.

The online uploaded file must be a ZIP file called **assignment2_STUDENTNR.zip**, containing multiple files and directories. The contents of the zip file are specified below (**bold** text indicates folder names):

- STUDENTNR_assignment2_report.pdf
- **section1_code**
 - *[your code files]*
- **section1_images**
 - *[160 .csv files with the following naming format: STUDENTNR_LABEL_INDEX.csv]*
 - *Optionally, the PGM files used to create the csv files, with the same naming format*
- **section2_code**
 - *[your code files]*
- **section2_features**
 - STUDENTNR_features.txt

- **section3_code**
 - *[your code files]*

A RAR file is not a ZIP file. A broken or corrupt ZIP file is not a ZIP file.

It is your responsibility to ensure the assignment is uploaded and double-checked before the deadline.

Please use the provided report template for preparing your report (or create an equivalent LaTeX format). Ensure that the header and footer information (student name, student number) is clearly visible on the PDF. The word limit for the report is 4000 words (excluding tables and figures; you can include as many tables and figures as you feel is appropriate).

By submitting this assignment you acknowledge that it is your own work and that you are aware of university regulations regarding academic offences, including (but not restricted to) plagiarism and collusion.

Standard university penalties apply for late submission.