

Lab08 - ML: Machine Learning

[Implement in PC, no need of VM]

Demo: 01, 03 April

Submission: 01, 03 April

Deliverables: submit files to Vocareum

Files to submit:

- train.ipynb , model.h5
- README.md

Objectives are to facilitate students understand -

- Introduction to Jupyter Lab
- Machine Learning
- Supervised vs Unsupervised Learning
- Regression vs Classification

This Lab is split into 3 parts:

First, an introduction to JupyterLab, be sure to follow the installation instructions. This will allow you to open the demo jupyter notebooks, which will come in handy for completing your task.

Secondly, an overview of machine learning is presented. Here you will be asked some questions as you go through each section. Answer the 5 questions in the README.md.

Lastly, you are going to implement a classification model using ANNs. To do this, completing the two previous parts will be extremely helpful. **Don't jump straight into this section** because the task context is given as a running example in the machine learning overview.

Introduction to Jupyter Lab

In this lab, we are going to be working with the [JupyterLab](#). It is a nice environment to quickly write, modify and run your code.

In addition to JupyterLab, you would also be needing a few python libraries to work with csv files, numbers/matrices, data and plotting. Some of them are **numpy**, **pandas**, and **seaborn**. On your local machine, inside the lab directory, run the following to install the required python libraries.

```
sudo apt-get install requirements.txt
```

OR

```
pip install -r requirements.txt
```

[Don't blindly copy]

Look for the JupyterLab tutorial video in the same lab folder as this document on GDrive.

Play around with it till you get the hang of it.

Machine Learning

Machine Learning is a branch of Artificial Intelligence which uses data and algorithms to allow computers to make decisions. Just like with humans, the computer is trained with some amount of data before it can start making accurate predictions. You and I learned to identify animals such as puppies by looking at multiple different examples and remembering key features. Machines can do the same thing with the training data supplied to them.

There are three main subcategories of machine learning; *supervised* learning, *unsupervised* learning and *reinforcement* learning.

Supervised learning: Here we “supervise” the machine's learning by providing it with the right and wrong information - the training data is often

labeled. If we use our puppy example, we would provide the computer multiple images where some of them would be labeled as puppies and others would not. The machine would then learn ways to identify pictures of dogs on its own.

Unsupervised learning: In unsupervised learning the machine is provided with unlabeled data and it looks for common and hidden patterns in the data the humans might not even realize exist! There is, however, some form of human interaction needed in order to validate the output data. This form of learning is sometimes used when there is a lot of information that even humans cannot easily classify but computers, being fast and good at crunching numbers, can.

Reinforcement learning: Reinforcement learning is another way machines learn exactly like humans do - by making mistakes! In this subcategory, machines make decisions and then are told whether their decision was right or wrong. For example, if the machine is shown a picture of a labrador puppy, and it guesses the puppy, the machine is rewarded (with some score), and the machine will learn to identify similar pictures as puppies. On the other hand, if the machine is shown a picture of a raisin muffin and it identifies it as a puppy, the machine will not be rewarded (or maybe negatively rewarded, i.e. punished), and it will learn to identify similar pictures as not puppies.

Supervised vs Unsupervised Learning

As mentioned above, the main difference between supervised and unsupervised learning is the inclusion of labels for the data vs unlabeled data. Due to this difference, there are differences in the machine learning algorithms (classification and regression vs clustering, association and dimensionality reduction), but here we will focus on the benefits and drawbacks of each of these subcategories and which one you should use.

4 key differences are:

Goals: In supervised learning, the objective is to predict outcomes on new data. By teaching a machine how to classify pictures as puppies and not-puppies, we want our algorithm to be able to correctly classify a new picture. In unsupervised learning, the objective is to analyze and draw patterns from large volumes of data. The computer itself will determine the differences and similarities between the data.

Applications: Thus, supervised learning is ideal for tasks such as spam detection and weather forecasting etc. while unsupervised learning is better suited for recommendation engines and medical imaging etc.

Complexity: Supervised learning is generally simple and we generally do not need a lot of computing power to run supervised learning algorithms. On the other hand, unsupervised learning algorithms are usually very complex as they have to parse through extremely large datasets and extract key features.

Drawbacks: Supervised learning can be time-consuming to train, especially given humans will manually need to add labels, while unsupervised learning methods may have inaccurate results.

Regression vs Classification

Classification refers to the process of grouping data into different categories. It is an example of [supervised learning](#). A simple example of classification would be, given a set of emails, separate them into two different categories, spam and non-spam. To learn more about classification feel free to explore [this link](#)!

Regression refers to the process of predicting the relationship between multiple variables. It is also an example of supervised learning. A simple example of regression would be, given a set of land parcel sizes, predict their value. The simplest form of this problem would NEGLECT other

factors such as location, zoning, etc., so you would get a linear regression where, as the land size increases, the value increases. However, a more complex regression can be created where those other factors are taken into account.

As you can see, both these concentrations of machine learning take some data and predict some output, however, it is extremely important to note the difference between them. Classification returns a discrete output while regression returns a continuous output. Intuitively this makes sense, because classification sorts data into categories, and you can either be in one category or another, but not in multiple categories at the same time right? Regression, on the other hand, is not so black and white. For example, in the land price example stated above, our price could be continuous value, rather than just discrete values (like only 10k\$, or 20k\$, or 30k\$, etc).

In this lab, we will focus on Supervised Machine Learning for Classification.

Classification

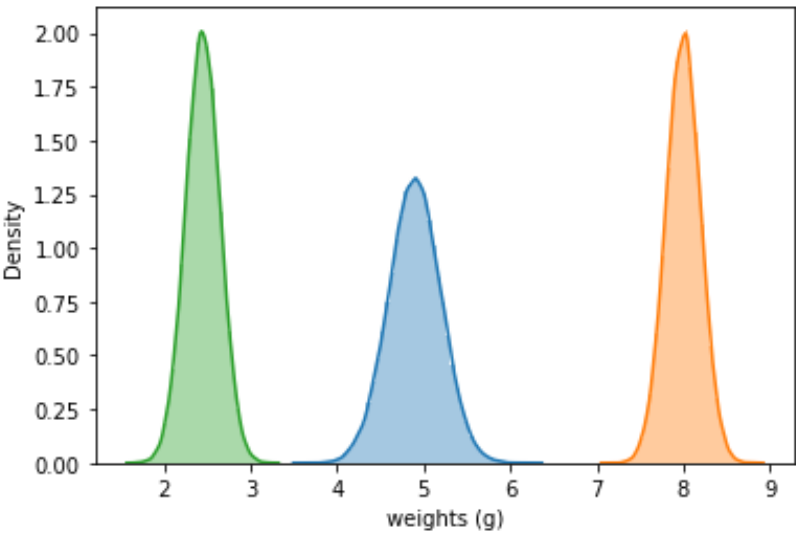
Motivation:

You work for a company that designs vending machines for their customers. A vending machine needs the ability to classify coins put into it. Here in the United States, vending machines might just use the coin diameter or the weight to classify coins. (Although that is not fool-proof, since one could put arbitrary metal coins of similar weights and diameter in an attempt to spoof the vending machine - but we are going to assume that all Americans are honest and good people.)

Source : [Coin Specifications | U.S. Mint \(usmint.gov\)](https://www.usmint.gov/coin-specifications)







Let us look at the following distribution of weights of 300,000 US coins of 3 denominations (100,000 coin samples of each denomination). The x-axis is

weights
y-axis is
estimated
density of







and the
the
probability
the 3

denominations.

	Cent	Nickel	Dime	Quarter Dollar	Half Dollar	Dollar
Denomination						
Composition	Copper Plated Zinc 2.5% Cu Balance Zn	Cupro-Nickel 25% Ni Balance Cu	Cupro-Nickel 8.33% Ni Balance Cu	Cupro-Nickel 8.33% Ni Balance Cu	Cupro-Nickel 8.33% Ni Balance Cu	Manganese-Brass 88.5% Cu 6% Zn 3.5% Mn 2% Ni
Weight	2.500 g	5.000 g	2.268 g	5.670 g	11.340 g	8.1 g
Diameter	0.750 in. 19.05 mm	0.835 in. 21.21 mm	0.705 in. 17.91 mm	0.955 in. 24.26 mm	1.205 in. 30.61 mm	1.043 in. 26.49 mm

Question 1: What are the denominations of the US coins from the green, blue and orange distributions? Can you think why the coins from the same denomination might show variation in weight although they are specified to be of the same weight? If your vending machine had a weight sensor, how would you use the weight of a coin that was just inserted to find the denomination?

Now, we want to consider a slightly different scenario: **Indian Rupee coins.**

Denomination	Metal Weight Shape Size	Obverse	Reverse
Ten Rupees	Bimetallic Cupro-Nickel in Center Aluminium Bronze in outer ring 7.71 gm Circular 27 mm		
Two Rupees	Ferritic Stainless Steel 5.62 gm Circular 27 mm		

Denomination	Metal Weight Shape Size	Obverse	Reverse
Five Rupees	Cupro-Nickel 9 gm Circular 23 mm		
Two Rupees	Cupro-Nickel 8 gm Circular 28 mm		
Two Rupees	Cupro-Nickel 6 gm Eleven sided 26 mm		

Source: [Reserve Bank of India - Museum \(rbi.org.in\)](http://rbi.org.in)

As you can see, different denominations may have the same size and same denominations may have different sizes/weights. So just using one feature like the diameter or the size is not going to be enough to determine the denomination.

Question 2. We can shine light on the coin and measure the reflected amount of light which should be proportional (directly or in some non-linear way) to the size/area of the coin. Can you guess which sensor on the Grove Pi Kit can be used for this purpose?

Since the coins are made up of different metals and have different thickness, we can measure their weight to see if they help us in making a decision to determine the denomination. We can use a weight sensor (something like this - [Gravity: Arduino Digital Weight Sensor -DFRobot](#)) to measure the weight of the coins.

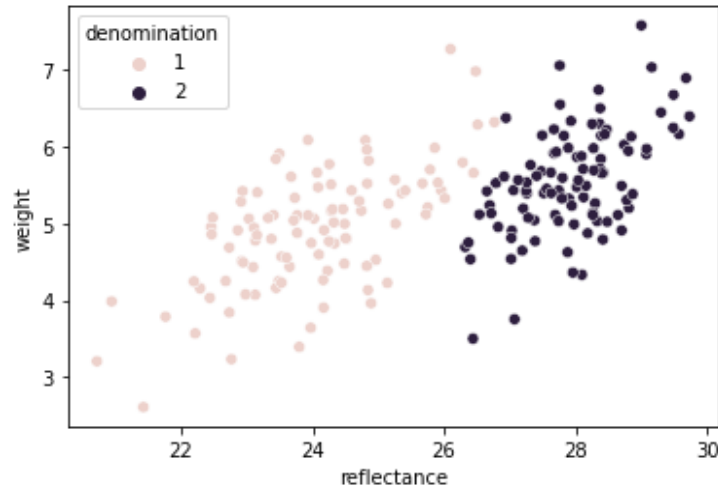
Main Doable:

Completing the **train.ipynb** file in the GDrive will complete the task for this lab. Make sure the coins.csv is in the directory where you open the ipynb file.

Problem Statement [USE coins.csv as your dataset for train/test]

In this lab, you will be given a set of “labeled data,” i.e. a set of reflectivity and weight values for coins of known denominations.

Here is the distribution of the coin reflectance (in some units) and weights(in grams) along with their labels(denominations).



We want to use this dataset to train a classifier/ML model that can predict coins that we might not have seen yet. For example, if you built and installed this vending machine, and new coins are introduced next year with new sizes and weights, the machine still needs to be able to detect/classify the correct denomination. The property of a model to be able to predict classes (denomination here) of data it has never seen (i.e not seen during training of the model) is called **generalization**. Without generalization capacity, a model is just memorizing since it can only predict about the data it has already seen (this is similar to students being able to solve problems in an exam only if they have seen the exact problem before - they are memorizing, and not learning since learning means to be able to **generalize**).

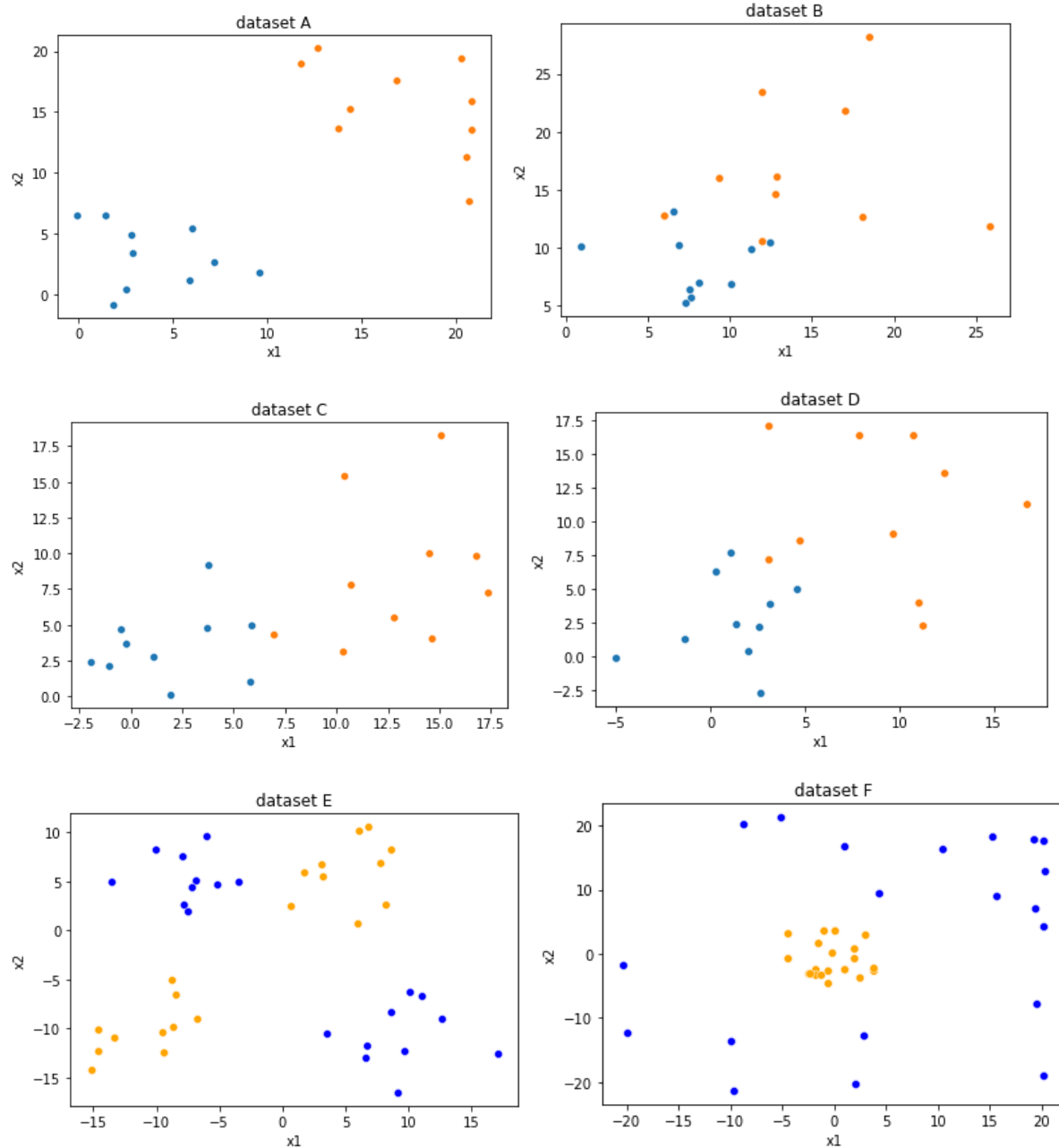
Although it is theoretically possible to work out a set of boundaries and rules manually (you can try, but it might not be fun), we will be using **Machine Learning (ML)** to derive the boundaries for the classification - called the decision boundary - as this boundary helps us decide which class an unseen datapoint should be predicted to belong to.

Linear Separability and Separating Hyperplanes

Sometimes, the data points belonging to the different classes are separable by a single hyperplane (a plane in 3D or a line in 2D or a hyperplane in

4D+). This means that we can draw a hyperplane (or a line in 2D) such that all the points belonging to one class fall on one side of the hyperplane and the data points belonging to the other class fall on the other side.

Question 3: Which of the following datasets are linearly separable? Justify your answer.



Question 4. Sometimes we need more than a simple hyperplane to separate the datasets of the two classes. What are some other simple

geometric entities other than a simple plane/line that can be used to separate some of the data points that were not linearly separable?

Hints: Lookup the following equations and find out what they represent

$$(x - h)^2 + (y - k)^2 = r^2$$

$$(ax + by + c)(Ax + By + C) = 0$$

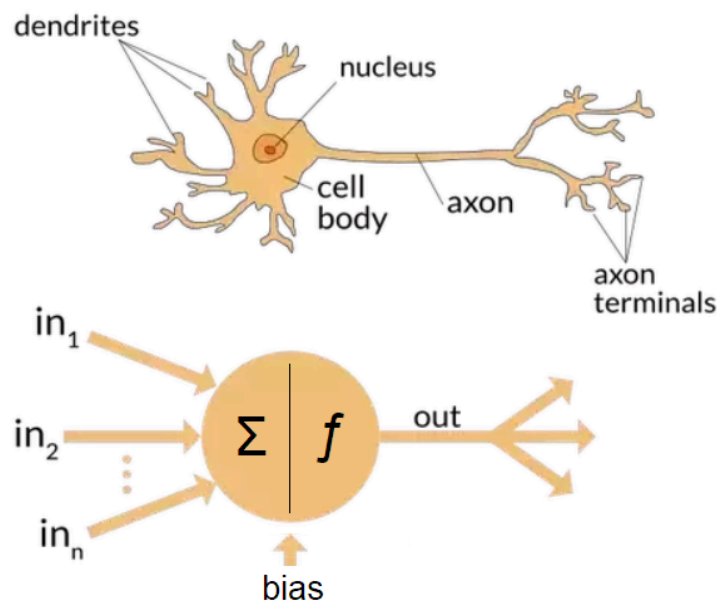
$$(y - k)^2 = 4a(x - h)$$

$$\frac{(x - h)^2}{a^2} + \frac{(y - k)^2}{b^2} = 1$$

$$(x - h)(y - k) = c$$

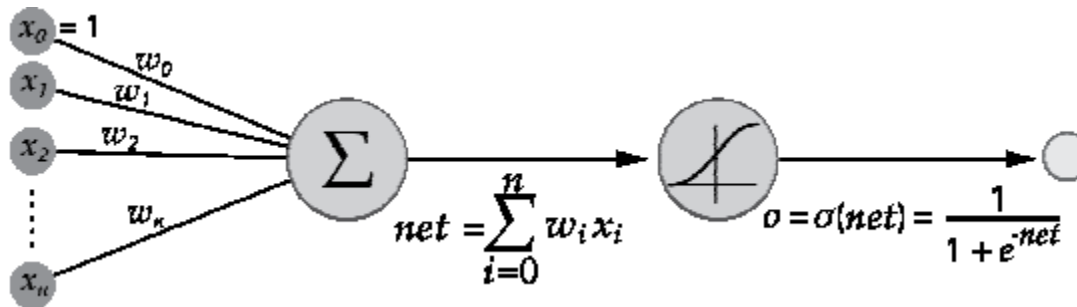
Artificial Neural Networks (ANNs)

Artificial Neural Networks (ANNs) were designed initially to mimic the neuronal networks found in sophisticated living beings (Biological Neural Networks - BNN).



Neural networks are one of the most powerful and complex machine learning models although each neuron is relatively simple to study. Each neuron is very much like a perceptron except that the threshold/step function might be replaced by any general function - mostly a nonlinear function called the **activation function**. A step function is nonlinear but it kind of does not provide any information about how far we are from the optimum because of its flat/horizontal shape which makes its derivative/gradient vanish. Smooth and non horizontal shaped nonlinear

functions are more suitable and are easy to train (due to non zero derivatives).



As you can see in the above picture, the single neuron here is basically a perceptron but with a sigmoid activation function instead of a step function. A perceptron is a special case of an artificial neuron where the activation function is the Heaviside step function.

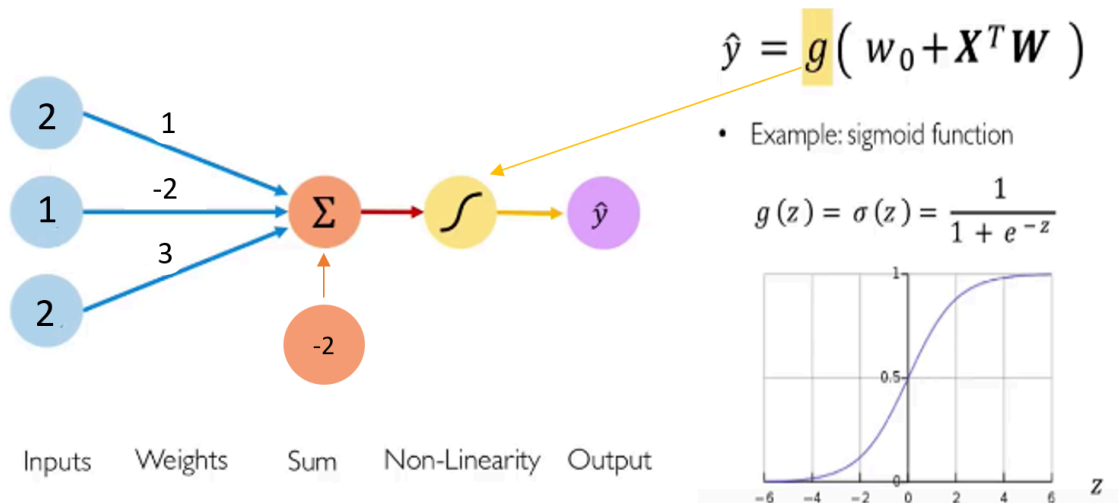
The most popular and successful algorithm to train an ANN is the backpropagation algorithm although it is not the only algorithm that can be used to train ANNs.

We will not go into the details of the training and working of the ANN in this lab but you will always get a chance to see them in any machine learning course since they are too powerful to be not visited.

The only thing we mention is the output of a single neuron can be modeled as -

$$y = f(W\vec{x} + \vec{b})$$

Here, y is the output of a neuron, the W matrix is called the weights of the neuron and the vector b is called the bias of the neuron. When training a neural network, we are trying to learn the weights and biases associated with the different neurons in the network so that together they are able to predict the correct classes of unseen data.



Question 5. For the example shown in the figure, what is approximately the output of the neuron?

the bias is -2[if anybody has any confusion]

You can watch these amazing videos which show the animations of an ANN at work and give explanations about their behavior.

<https://www.youtube.com/watch?v=UOvPeC8WOt8>

<https://www.youtube.com/watch?v=3JQ3hYko51Y>

<https://www.youtube.com/watch?v=oJNHXP0XDk>

The code for training ANN is present in the examples folder. Highly recommended to go over it to see how easy it is to train an ANN using available libraries.

Please go over this tutorial to learn about implementing a NeuralNetwork using Tensorflow and Keras -

<https://machinelearningmastery.com/tutorial-first-neural-network-python-keras/>

Very simple visual representation of the above Keras/TensorFlow code tutorial (for anyone who would like further elaboration): [EE 250 - ML Lab 7: Simple Visual Clairification](#)

1. Demo and Code Grading Rubric

All files are to be submitted via Vocareum in teams.

<u>Points</u>	<u>Description</u>
<u>DEMO</u>	
5	Accurate explanation on how the test and training data are going to be used to train the ANN classifier.
5	Show plot of decision boundary with sample points.
<u>Code</u>	train.ipynb, model.h5
1	List all team member names in the README.md
2	Test and train data splitting
2	Correct implementation of the classifier on the training data
3	Decision boundary plotted
2	Calculate training and testing accuracy
<u>Questions</u>	README.MD
10	Answer questions Q1 to Q5 (2 points each)
	Total Possible: 30