**Classification**

* Predicting a discrete value, which could be one of n number of options
* Example, predicting if an object detected is metal, plastic, or another material

**Regression**

* Predicting a continuous value, which could be any number.
* Example, predicting the price of a house in a given area.

**Linear Regression**

* Finds the line of best fit amongst the entropy.
* Entropy is the level of disorder or uncertainty in a given dataset
* Equation of a line is as follows.
* Y = y coordinate
* m = The slope of the line (In machine learning, we typically refer to the slope as the weight)
* x = x coordinate
* b = y intercept
* If m is equal to 220, it means that for each increase in x, y increases by 220.
* In machine learning, we may see the same equation denoted differently
* Y = y coordinate
* = y intercept
* = Slope of the line
* = x coordinate
* This is fine if there are just two variables, however, we may wish to have more. For example, to calculate the price of a house, we may have the variables of square footage, number of bedrooms and number of bathrooms. We can add more variables to the soup like such.
* = Square Footage Variable
* = Number of Bedrooms Variable
* = Number of Bathrooms Variable
* = All beta values are the weights for its associated variable
* Does not react well to outliers
* Only models linearity

**Decision Trees**

* Supervised learning model for both classification and regression
* Pretty much large if/else statements, where the answer to the previous if/else affects which if/else will be asked next.
* Prone to overfitting

**Random Forest**

* Collection of unique decision trees
* Each tree makes its own decisions, which are then classified. The result from each tree is then compared amongst all other trees.
* For classification problems, the result with the highest “votes” is the final prediction
* For regression problems, the results are all averaged together to get the final predicted average
* Less sensitive to the training data, compared to decision trees
* Overfitting is still a concern
* The depth of a tree is the maximum number of questions asked before it makes a prediction. If we limit the depth of the tree, we limit the chance of overfitting, but may increase our chances of underfitting

**Gradient Boosting**

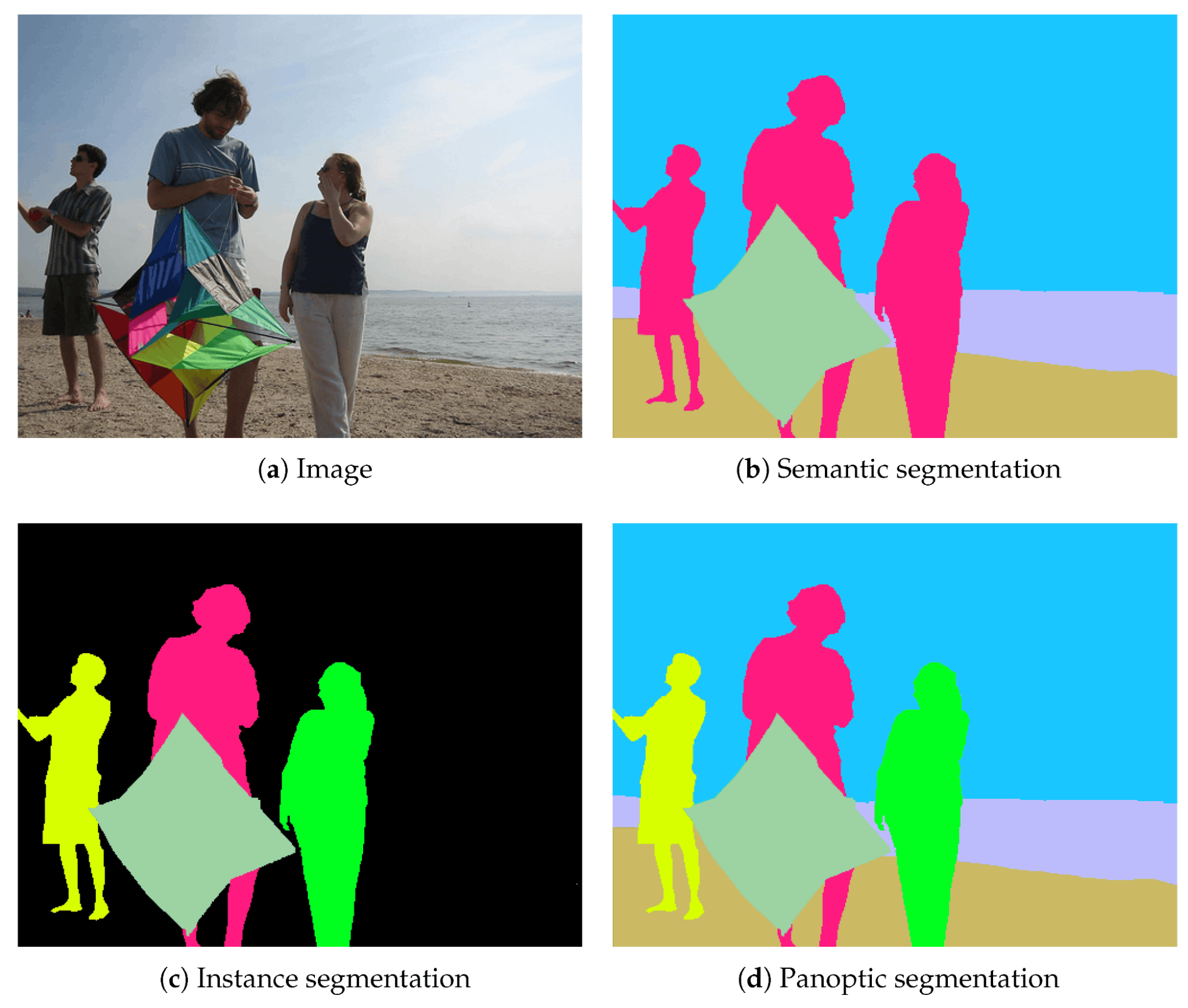
* Also a collection of decision trees, difference between gradient decent and random forest is how the trees are generated and how they are aggregated
* Each tree is made one after another, building upon the efficiencies of the previous tree (This is called boosting)
* Results are also aggregated along the way, instead of at the very end
* Typically a better than random forests, but are still prone to over fitting
* Also sensitive to outliers, so not a good idea if there is a lot of noise in the data
* XGBoost is a popular library of highly optimized gradient boosting.
  + Helps with cross validation
  + Missing data
  + Cross-validation
  + Parallelization
  + Tree pruning (making trees deeper, but more optimised)

**Genetic Algorithms**

* Based on the concept of evolution
* Uses natural selection to approximate solutions for given problems
* Generation 0 has a population made up of completely random variables
* A fitness function is used to gauge the effectiveness of each population member
* The two population members with the highest fitness score is then used as a template for the next generation, where a new population is made (cross-over function)
* Like gradient boosting, it builds on previous versions
* Elitism is when we take the population member with the highest scoring fitness and put it into the next generation unchanged (it survived in the game of survival of the fittest) This makes sure that if no better population member can be made, we do not accidently overwrite the best
* This is then continued until we are satisfied with the performance of a given generation, or until a predetermined number of generations has been met
* Some examples may include simulations of robots walking, maze solvers, colour adapters, etc…
* Genetic algorithms show similar traits to reinforcement learning, but it is not the same.
* This makes blind coordination and combinations, without understanding or meaning. Combining genetic algorithms with Neural Networks can help with this.

**Image Segmentation**

* Image segmentation is about dividing up an image into distinct sections, this can be done in various ways.
* Drawing a bounding box (rectangle) around a detected object is a form of linear regression, as you are finding the “best fit” for the xPos, yPos, length and height of the rectangle, to hold the maximum amount of the object, while maintaining the least amount of other objects.
* Other methods can be pixel accurate.
* a. original image
* b. semantic segmentation. Each class is assigned a specific colour
* c. instance segmentation. Each instance of a specified class is assigned a specific colour
* d. Panoptic segmentation. Each instance of each class is assigned a specific colour



* The above methods are very computationally expensive, can take seconds to minuets per frame to render. Big issue if time critical (real time self-driving cars), less of a problem for non-time critical issues (medical image highlighting on a single image)