1)

a)

IG(D, house) = 0.2420

IG(D, trance) = 0.0756

IG(D, dutch) = 0.1043

IG(D, detroit) = 0.0718

(house) - acid -> (trance) - goa -> (+)

(house) - acid -> (trance) - beat -> (-)

(house) - acid -> (trance) - psycho -> (+)

(house) - electro -> (detroit) -> yes -> (+)

(house) - electro -> (detroit) -> no -> (-)

(house) - deep -> (-)

b) Nearest neighbours are 1, 7 and 12. Majority = ‘-’  
  
Distance func = 0 if same else 1

Weight func = 1/x

c) Not examined

2)

Not assessed anymore

3a)

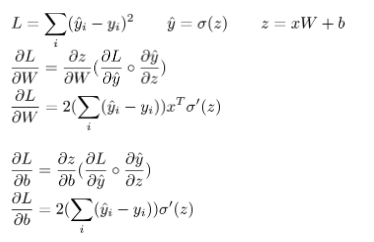
Using this addon for the latex: <https://gsuite.google.com/marketplace/app/autolatex_equations/850293439076>

￼

￼

￼

￼

￼

b) Vanishing gradients. Use cross entropy loss.

Attempting to explain the above, not sure if this explanation is correct but sigmoid constrains y\_predict to 0 <= y\_predict <= 1 which means that abs(y\_predict – y) <= 1 always. Combined with derivative of sigmoid which is also < 1 always causes the gradient to be very small and as we propagate the gradient towards the input layers this update amount becomes tiny and nothing is learned which means vanishing gradients.

c) Dropout is a technique where during training some activations are randomly zeroed. This is a form of regularisation, and prevents the network from relying on any one node. It is typically set to 50% for hidden layers. The activations will be 50% smaller, so they need to be scaled up during test-time.

Also: it is common to use a higher retention rate / lower dropout rate for input nodes ~20%

d)

Class 1 Recall: 4/5

Class 1 Precision: 1

Class 1 F1: 8/9

Class 2 Recall: 1

Class 2 Precision: ⅙

Class 2 F1: 2/7

Unbalanced classes, 500 vs 20 class size.

Metrics sensitive to class size. Use alternatives e.g. ROC-AUC. Or Downsample the data

Normalise confusion matrix