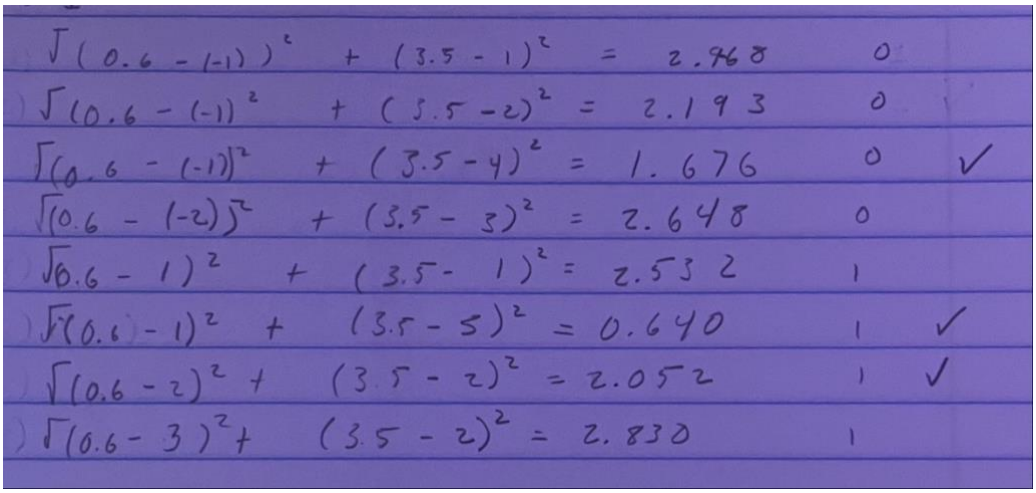
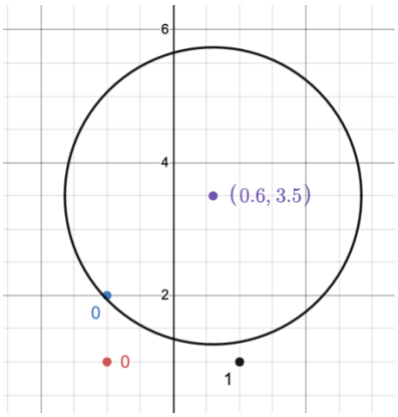
CSCI 4380 | HW 1
Ruben Cristea
811427744

1)



1. No.9 would be classified as 1
2. 

3. The first disadvantage is that data in the training set can have missing and outlying data that can affect the value of the variable. For instance, if all the data is accepted from parts 1 and 2 with k=1, we will classify No.9 as 1 but in 2, we used only three data points in our training set and classified No.9 as 0.

   The second disadvantage is that with huge datasets, calculating the distances for each point is very computationally heavy, slowing down the performance.

2)

GradientDescent

```
#1 sum((w.dot(x) - y + w_0)**2 for x, y in points) / len(points)
#2 return [
      sum(2*x*(w.dot(x) - y + w_0) for x, y in points) / len(points),
      sum(2*x*(w_0 *x[0] - y) for x, y in points) / len(points)
    ]
#3 w_0 = w_0 - eta * gradient_w_0;
```

iteration 0: w = [0.31333333], w_0 = [0.31333333],F(w, w_0) = 8.75
iteration 1990: w = [0.5588585], w_0 = [0.81034483],F(w, w_0) = [1.79558407]
iteration 2000: w = [0.5588585], w_0 = [0.81034483],F(w, w_0) = [1.79558407]

StochasticGradientDescent

```
#1 return ((w.dot(x) + w_0) - y) ** 2
#2 return [
         2 * ((w.dot(x) + w_0) - y) * x,
         2 * (w_0 * x - y) * x
       ]
#3 w_0 = w_0 - eta * gradients[1]
```

iteration 0: w = [0.13243344], w_0 = [0.13455168],F(w, w_0) = [7.78640588]
iteration 5900: w = [0.55768373], w_0 = [0.8070158],F(w, w_0) = [1.89052214]
iteration 6000: w = [0.54977978], w_0 = [0.79956639],F(w, w_0) = [1.8945029]

3)

1. Since the probability of yes is higher than the probability of no, then if it sunny, mild, and high temps they will play golf. So 9 will be yes.

	Yes	No	P(Yes)	P(No)
Sunny	1	2	0.25	0.50
Rain	1	2	0.25	0.50
Overcast	2	0	0.50	0.00
Total	**4**	**4**	**1.00**	**1.00**
Hot	1	2	0.25	0.50
Mild	1	0	0.25	0.00
Cool	2	2	0.50	0.50
Total	**4**	**4**	**1.00**	**1.00**
High	2	2	0.50	0.50
Normal	2	2	0.50	0.50
Total	**4**	**4**	**1.00**	**1.00**

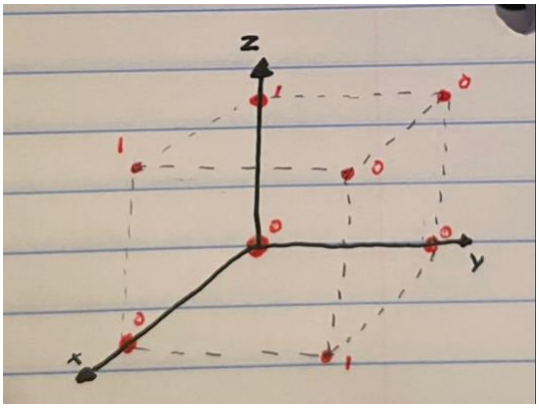
Sunny & Mild & High:
Yes – 0.25 * 0.25 * 0.50 = 0.031
No – 0.50 * 0.00 * 0.50 = 0.00

2.
Training: O(# instance * # of options for each feature) so O(n)
Testing: O(# of classes * # of options for each feature) so O(1)
3. (0.00005 * 0.98) / ((0.00005 * 0.98) + (0.99995 * 0.02)) = 0.00244
There is a 0.244% chance the developer has the disease.

4)

1. We want to split the data for training and testing, but we also want to maximize the data we can use for both. So, we split the data into k subsets and choose 1 to be testing, then the rest to be training. We then do k separate trainings for each subset. This was all the data can be used for testing and training.
2. Yellow is most likely overfitting while red is most likely underfitting.
3. As k increases, overfitting is less likely to happen and the model will start to underfit
If k is small, slight changes in data will more likely change the target data, causing the model to overfit.

5)



No. Based on the graph, there is no possible way to separate the data classified as 1, from the data classified as 0 using a single plane. So a single perceptron will not be able to solve this classification problem.

…

6)

$$d(x, y) = \sqrt{\sum_{i=1}^{D} (x_i - y_i)^2}$$

Suppose that $D = 2$ so, suppose:

$$\sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2} \leq \sqrt{x_1^2 + x_2^2} + \sqrt{y_1^2 + y_2^2}$$

$$\left(\sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2}\right)^2 \leq \left(\sqrt{(x_1^2 + x_2^2)} + \sqrt{(y_1^2 + y_2^2)}\right)^2$$

$$(x_1 - y_1)^2 + (x_2 - y_2)^2 \leq 2\sqrt{x_1^2 + x_2^2}\sqrt{y_1^2 + y_2^2}$$

$$-2x_1y_1 - 2x_2y_2 \leq 2\left(x_1^2 + x_2^2\right)\left(y_1^2 + y_2^2\right)$$

$$x_1y_1 - x_2y_2 \leq \left(x_1^2 + x_2^2\right)\left(y_1^2 y_2^2\right)$$

Cauchy States

$$\sum_{i=1}^{n} x_i^2 \sum_{i=1}^{n} y_i^2 \geq \left(\sum_{i=1}^{n} x_i y_i\right)^2$$

∴ since euclidean can be reduced to cauchy inequality, euclidean distance is a valid distance metric