1) a) 3 support vectors: (0,1), (-2,-1), (0.5,-0.5)

b) $d = \dfrac{|g(x)|}{\|w\|} \Rightarrow$

$$\frac{|0+1|}{\sqrt{1^2+(-1)^2}} = \frac{1}{\sqrt{2}}$$

$$\frac{|-2+(-1)|}{\sqrt{1^2+(-1)^2}} = \frac{3}{\sqrt{2}}$$

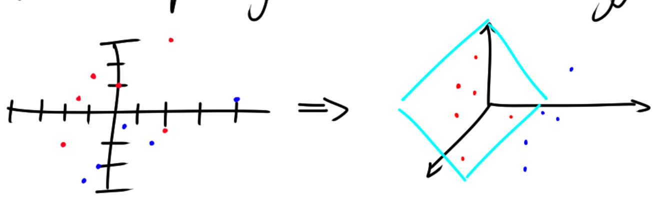
$$\frac{|0.5+(-0.5)|}{\sqrt{1^2+(-1)^2}} = \frac{1}{\sqrt{2}}$$

c) If the sample (0,1) is removed, the decision boundary will change because it's a support vector – which defines the decision boundary / hyper-plane.
If the sample (-1, -2.5) is removed, the decision boundary will not change because it's not a support vector.

d) If the positive sample (2, 0.5) is added, the decision boundary will change because it would no longer be linearly separable.
To handle this, we could use the following methods:

- Slack Margin Classification, where we would introduce a slack variable ξ to allow misclassification of noise – resulting in soft margins.

- Non-linear SVMs – provided that the above method doesn't work efficiently, we could transform the space to a higher dimension, map the originals to the new dimension accordingly, and separate by some plane in space.



e) Very large values of C means that error really matters – meaning that any errors would make the equation big (generally, we don't want to see that). i.e. The weight of the errors are proportionally larger the larger the value of C is.

Very small values of C would mean we only care more about the margin, and not the errors (or when $C=0$, we don't care about it at all), i.e. the weight of the errors are proportionally smaller the smaller the value of C is.

f) **Hard Margin** vs. **Soft Margin**

Hard Margin:
- Doesn't tolerate outliers – requires a hard linear separation/margin b/t classes
- Outliers affect boundaries
- Dealing w/ very distinctive data/classes:
  - Plane vs. Boat
  - Dog vs. Cat

Soft Margin:
- Provides a principled way to handle noisy data/outliers
- Dealing w/ somewhat similar, but distinctive data/classes:
  - The shapes ☆ vs. ☆
  - An SUV vs. a Van.
  - Planets

Linear vs **Kernel**

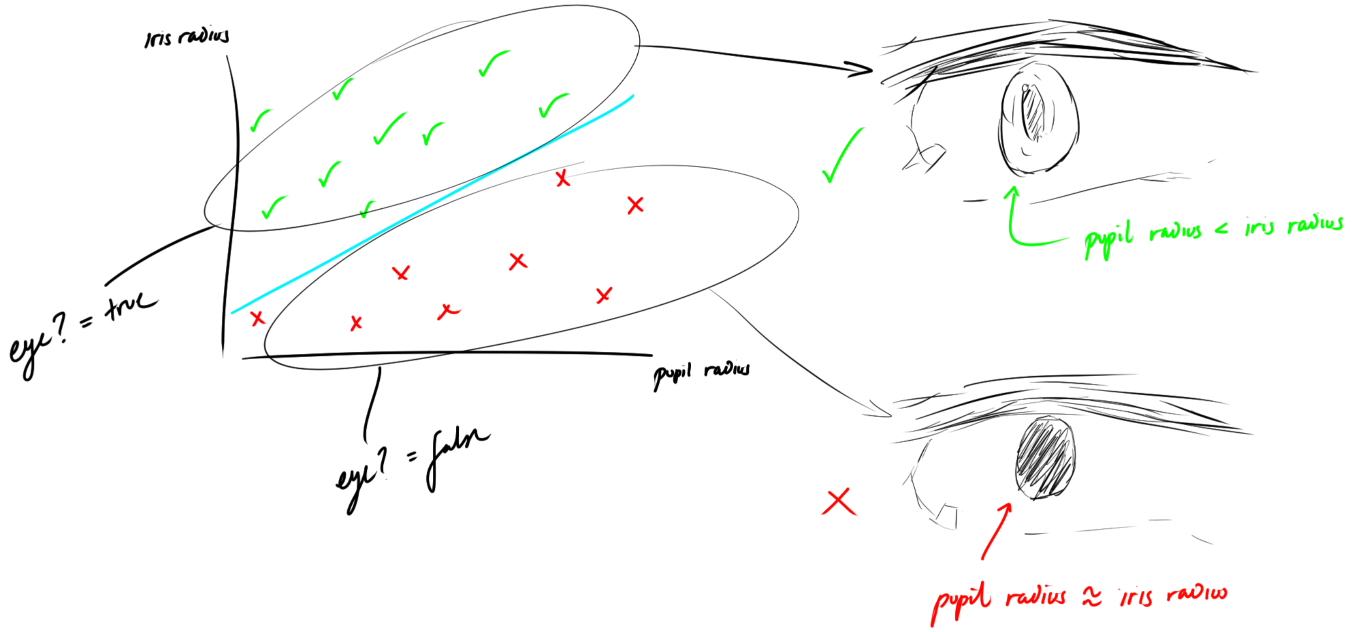
Linear:
- When working w/ linearly separable data
- Classify b/t 2 generally distinct sets of data/classes:
  - Digits vs. Letter
  - Dog vs. Cat

Kernel:
- non-linear data
- general use
- e.g. Computer Vision & Graphics:
  - Facial recognition, where colors in an image vary ⇒ non-linear

1)f)

When it comes to real-world applications, one would generally want to choose a method that isn't vulnerable to noise when it's introduced, but instead allows it with some weight/tuning parameter. So when it comes to hard vs soft margins, I'd go w/ soft b/c it provides a principled way to handle noise. For example, let's say that we're working with classifying apples and oranges based on color and shape. Some species of apples are yellow and not red, or could be as round as an orange - meaning that they would cluster w/ the data that represents oranges (i.e. they're outliers). Hard margins wouldn't work w/ these problems, but soft would.

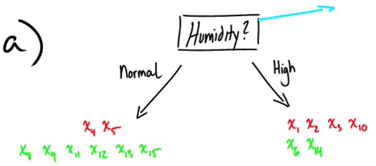
When it comes to linear vs kernel SVMs, I would choose linear SVMs when it comes to real life applications. Computer Vision, which is commonly utilized in real life for facial and image recognition, would benefit more from linear implementations rather than kernel. For example, a use case would be to classify eyes given certain parameters and labels. With this information, a decision boundary could be drawn to separate what is an eye and what's not w/o having to go to higher dimensions to do so.



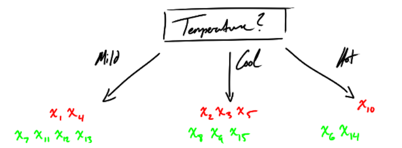
2)

Outlook	Temperature	Humidity	Run?	
Rainy	Mild	High	No	x_1
Overcast	Cool	High	No	x_2
Rainy	Cool	High	No	x_3
Overcast	Mild	Normal	No	x_4
Rainy	Cool	Normal	No	x_5
Overcast	Hot	High	Yes	x_6
Sunny	Mild	Normal	Yes	x_7
Sunny	Cool	Normal	Yes	x_8
Sunny	Cool	Normal	Yes	x_9
Rainy	Hot	High	No	x_{10}
Sunny	Mild	Normal	Yes	x_{11}
Rainy	Mild	Normal	Yes	x_{12}
Sunny	Mild	Normal	Yes	x_{13}
Sunny	Hot	High	Yes	x_{14}
Sunny	Cool	Normal	Yes	x_{15}

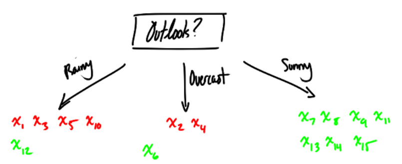
a)



$$I_H = -\left[\frac{9}{15}\left(\frac{2}{9}\log_2\frac{2}{9} + \frac{7}{9}\log_2\frac{7}{9}\right) + \frac{6}{15}\left(\frac{4}{6}\log_2\frac{4}{6} + \frac{2}{6}\log_2\frac{2}{6}\right)\right] \cong \boxed{0.826}$$

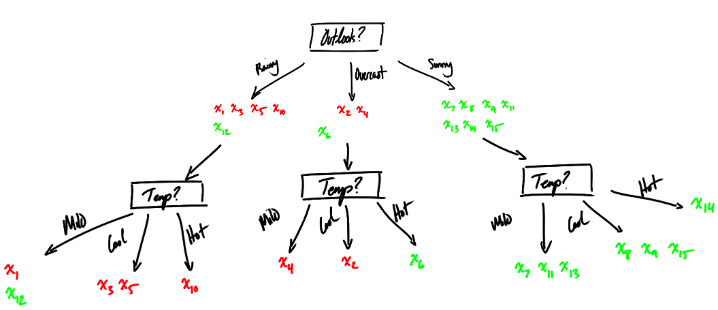


$$I_T = -\left[\frac{6}{15}\left(\frac{2}{6}\log_2\frac{2}{6} + \frac{4}{6}\log_2\frac{4}{6}\right) + \frac{6}{15}\left(\frac{3}{6}\log_2\frac{3}{6} + \frac{3}{6}\log_2\frac{3}{6}\right) + \frac{3}{15}\left(\frac{1}{3}\log_2\frac{1}{3} + \frac{2}{3}\log_2\frac{2}{3}\right)\right] = \boxed{0.951}$$

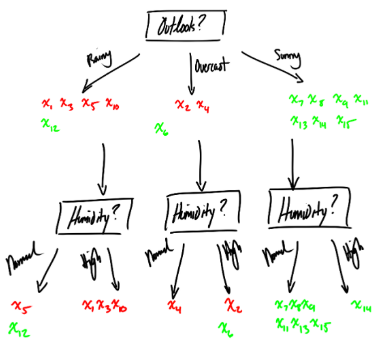


$$= -\left[\frac{5}{15}\left(\frac{4}{5}\log_2\frac{4}{5} + \frac{1}{5}\log_2\frac{1}{5}\right) + \frac{3}{15}\left(\frac{2}{3}\log_2\frac{2}{3} + \frac{1}{3}\log_2\frac{1}{3}\right) + \frac{7}{15}\left(\frac{7}{7}\log_2\frac{7}{7}\right)\right] = \boxed{0.424}$$

$I_O < I_H < I_T$ $\therefore$ split w/ Overcast



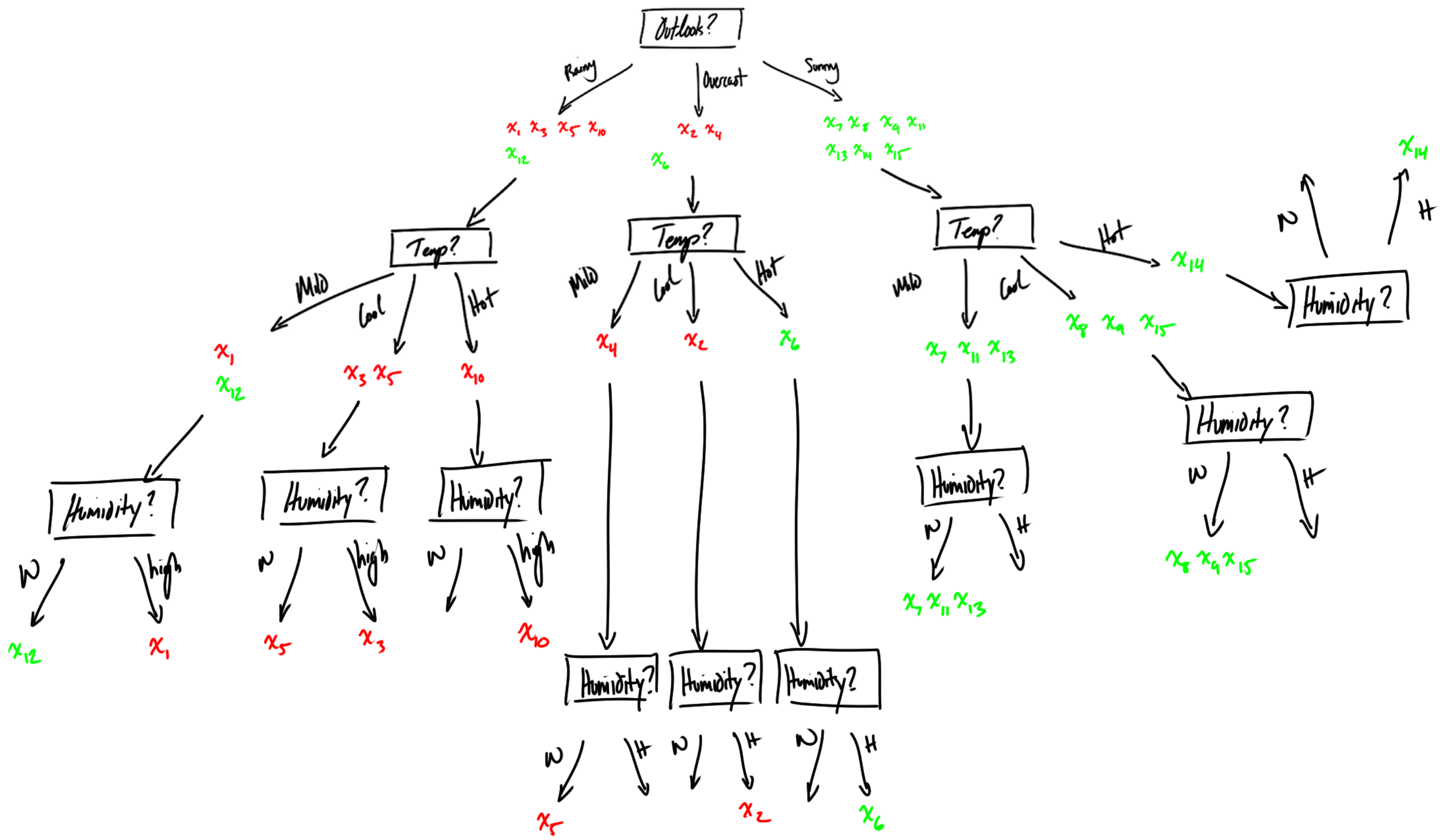
$$I_{OT} = -\left[\frac{2}{5}\left(\frac{1}{2}\log_2\frac{1}{2} + \frac{1}{2}\log_2\frac{1}{2}\right) + \frac{2}{5}\left(\frac{2}{2}\log_2\frac{2}{2}\right) + \frac{1}{5}\left(\frac{1}{1}\log_2\frac{1}{1}\right)\right] \quad 0.400$$
$$-\left[\frac{1}{3}\left(\frac{1}{1}\log_2\frac{1}{1}\right) + \frac{1}{3}\left(\frac{1}{1}\log_2\frac{1}{1}\right) + \frac{1}{3}\left(\frac{1}{1}\log_2\frac{1}{1}\right)\right] \quad + \; 0$$
$$-\left[\frac{3}{7}\left(\frac{3}{3}\log_2\frac{3}{3}\right) + \frac{3}{7}\left(\frac{3}{3}\log_2\frac{3}{3}\right) + \frac{1}{7}\left(\frac{1}{1}\log_2\frac{1}{1}\right)\right] \quad + \; 0$$
$$\boxed{0.400}$$



$$I_{OH} = -\left[\frac{2}{5}\left(\frac{1}{2}\log_2\frac{1}{2} + \frac{1}{2}\log_2\frac{1}{2}\right) + \frac{3}{5}\left(\frac{3}{3}\log_2\frac{3}{3}\right)\right] = 0.400$$
$$-\left[\frac{1}{3}\left(1\log_2 1\right) + \frac{2}{3}\left(\frac{1}{2}\log_2\frac{1}{2} + \frac{1}{2}\log_2\frac{1}{2}\right)\right] = + \; 0.\overline{667}$$
$$-\left[\frac{6}{7}\left(\frac{6}{6}\log_2\frac{6}{6}\right) + \frac{1}{7}\left(1\log_2 1\right)\right] = + \; 0$$
$$\boxed{\sim 1.\overline{067}}$$

$I_{OT} < I_{OH}$

$\therefore$ split w/ Temp



* Entropy of $1_{OTH} = 0$ b/c it's all pure & there's no other features to compare against.

b) The runner will NOT go for a run based on the decision tree and the inputs because based on the data, when it's overcast and cool, there is only 1 data point where it's a NO, ∴ humidity is disregarded in this specific case. However, one could say UNKNOWN b/c there needs to be more data to make a decision.

```
Training/validation accuracy for minimum node entropy 0.010000 is 1.000 / 0.863
Training/validation accuracy for minimum node entropy 0.050000 is 0.999 / 0.863
Training/validation accuracy for minimum node entropy 0.100000 is 0.997 / 0.865
Training/validation accuracy for minimum node entropy 0.200000 is 0.990 / 0.867
Training/validation accuracy for minimum node entropy 0.500000 is 0.963 / 0.863
Training/validation accuracy for minimum node entropy 0.800000 is 0.919 / 0.856
Training/validation accuracy for minimum node entropy 1.000000 is 0.871 / 0.840
Training/validation accuracy for minimum node entropy 2.000000 is 0.596 / 0.600
Test accuracy with minimum node entropy 0.200000 is 0.872
```

3) a) The most optimal theta to use in this case would be theta = 0.5 because although the training accuracy drops off noticeably, validation accuracy stays roughly the same until after the 0.5 mark. Obviously, the best theta to use is the lowest theta, but that would potentially incur higher computational costs.

b) What I could say about the model complexity of the Decision Tree given the training and validation accuracy is that it's somewhat complex because even with a minimum node entropy of 1.00, the training and validation accuracy is 0.871 and 0.840, respectively. This is because we are working with digits [0-9], and most of the digits have some semblance to one another, such as 2 and 7 (the long, slanted shaft), 0, 8, and 9, (the loops), and 6 and 9 (the loop and tail). Sure, decreasing the minimum node entropy increases training accuracy, but when tested against a validation set, the accuracy barely changes until theta = 0.5. And this is because, like I said, the similarities of certain digits and their features.