1. Decision Trees

(a) overall entropy: $-\sum_{i=1}^{c} P(x_i)\log_b P(x_i)$

$S = 5 \quad S' = 5$

$-\frac{1}{2}\log_2\left(\frac{1}{2}\right) = \frac{1}{2}$

$-\sum_{i=1}^{2} P(x_i)\log_b = \frac{1}{2} + \frac{1}{2} = 1$

$IG(T,A) = \text{Entropy}(T) - \sum_{v \in A} \frac{|T_v|}{T} \cdot \text{Entropy}(T_v)$

- Color:

R

S	S'
3	2

$\text{entropy} = -\left(\frac{3}{5}\log_2\left(\frac{3}{5}\right) + \frac{2}{5}\log_2\left(\frac{2}{5}\right)\right) = 0.97$

Y

S	S'
2	3

$\text{entropy} = -\left(\frac{2}{5}\log_2\left(\frac{2}{5}\right) + \frac{3}{5}\log_2\left(\frac{3}{5}\right)\right) = 0.97$

$IG = 1 - (.5(0.97) + 0.5(.97)) \approx \underline{0.03}$

- Type

sports $\frac{S \mid S'}{4 \mid 2}$ $e_1 = -\left(\frac{4}{6}\log_2\left(\frac{4}{6}\right) + \frac{2}{6}\log_2\left(\frac{2}{6}\right)\right) \approx 0.9183$

suv $\frac{S \mid S'}{1 \mid 3}$ $e_2 = -\left(\frac{1}{4}\log_2\left(\frac{1}{4}\right) + \frac{3}{4}\log_2\left(\frac{3}{4}\right)\right) \approx 0.8113$

$\frac{6}{10}e_1 + \frac{4}{10}e_2 \approx 0.55098$

IG $\approx 1 - 0.55098 \approx \underline{0.44902}$

- Origin

Domestic $\frac{S \mid S'}{2 \mid 3}$ $e_1 = e_2 = -\left(\frac{2}{5}\log_2\left(\frac{2}{5}\right) + \frac{3}{5}\log_2\left(\frac{3}{5}\right)\right)$

imported $\frac{S \mid S'}{3 \mid 2}$ (same as color split)

IG $\approx \underline{0.03}$

Best feature for first split is "Type" as it gave the most IG

(b) Split on type for the first branch. For each subsequent branch, repeat the same process as in (a) for each feature not yet used in that exact branch. Then the feature with the most calculated information gain. Repeat until branches or halting conditions are 0 entropy reached.

(c) We can stop when the number of cases in a node is less than some predifed limit, the purity of the node is more than some predifed limit, we have already split on the three features in every branch to (leaf node) or stop when the prediction values for all records are identical.

(d) No, there is no need in most cases to standardize the features when using a decision tree. Monotonic transformations do not affect the decision tree splits.

(e) Decision trees are robust to outliers as they split data into multiple groups rather than draw a hyperplane or logic weighted boundary.

2. True or False, Simple Explanations

(a) False, Boosting uses many learners with the idea of producing many cheaper weak learners is better than a few expensive strong learners

(b) False, at a split point in Bagging, the algorithm can look through all features to find optimal split point, while random forest is limited to random sample of features of which to search

(c) While both bagging and boosting decrease variance and lead to a model with high stability, only boosting can reduce bias. Also, bagging can be efficient in reducing overfitting, but boosting can increase overfitting and requires careful tuning on the hyper-parameters. Boosting can, however, support different loss functions.

(d) with certain activation functions such as the sigmoid function, the input is squished, so changes in the input lead to a small change in output. With many layers, this can cause a vanishing gradient in backpropogation. the ReLU is discontinuous, and the gradient is either 0 or 1, so gradients don't get squished. However, you can get ReLU's which return 0 forever, so never update/learn.

3. Overfitting Mitigation Strategies

3.1) Yes, a larger dataset most often is a better representation of population data, and less likely to generalize, so it will help avoid overfitting.

3.2) No, allowing the model to train for more iterations decreases bias and increases variance, as the model will be more fitted to the data and less generalizable with more iterations.

3.3) No, as more parameters leads to a more complex higher dimensionality model, which is more likely to overfit.

3.4) Yes, using this dropout method, we make the model learn some lost information in dropped nodes making it more generalizable and less prone to overfitting.

3.5) No, a specialized chip might be faster, but the process is the same and so is the risk of overfitting.

3.6) Yes, changing the initial values/strategy can lead to finding a local minimum instead of a global, leading ~~to~~ less overfitting as the model is less exact

4) Principal Component Analysis

(a) The advantages of dimensionality reduction are a removal of correlated features which lowers the input dimension and improves algorithm performance. It also reduces overfitting and improves visualization methods. However, independent variables become less readable and interpretable, data standardization must be done beforehand otherwise dimensionality reduction will lead to terrible results, and dimensionality reduction will lead to information loss.

(b)

- Data with an inherently linear structure will work well with PCA, as we can reduce dimensionality by taking the principal component along which the data lies, and due to low variation of other features, we can eliminate them.

- data lying on a hyperbolic plane will not work well with PCA, as unlike the other dataset, it is not obvious how to center points in hyperbolic space. As such, the steps in PCA do not work well with hyperbolic data.

- data containing unscaled features will not work with PCA, as without normalizing the data, we cannot be certain which features have more impact than the others

- statistically independent features indicate each feature provides information and should not be discarded, so PCA should not be used

5) Neural Networks

(a) All activation functions should provide nonlinearity and ensure gradients remain large enough (do not vanish) through the hidden layers. Common choices are sigmoid, tanh, ReLU and its variations, softplus, and swish. Input layers do not have activation functions, as the activation function is used in the forwards propagation of data and the input is the first step. Hidden layers can use ReLU(⌒), sigmoid(⌒), and tanh(⌒). Sigmoid and tanh can lead to a vanishing gradient issue, while ReLU prevents this. As such, I would use ReLU for the hidden nodes. The output activation function depends on the type of prediction that one is doing. If the target is binary, I would use a sigmoid function to get probability of class membership, if there is multiclass classification, then the output layer will have one node per class so a softmax activation should be used to help assign them. If it is a regression problem, just use a linear activation.

(b) $X_0 = -2$ $X_1 = 1$ $w_0 = 0.5$ $w_1 = 0.75$ $w_2 = -0.25$

bias term = 1

~~$p = \text{ReLU}(w_0 X_1 + w_1 X_2 + w_2(\text{bias})) = \text{ReLU}(-1 + \frac{3}{4}) = \frac{1}{4}) =$~~

~~$\text{ReLU}(-\frac{1}{4}) = \max(0, -\frac{1}{4}) = 0$~~

~~$q = \text{ReLU}$~~

$\text{hidden } 0 = \text{ReLU}(w_0 X_0 + w_1 X_1 + b) = \max(0, [-1 + \frac{3}{4} + 1]) = \frac{3}{4}$

$\text{hidden } 1 = \text{ReLU}(w_0 X_0 + w_1 X_1 + b) = \frac{3}{4}$

$\text{hidden } 2 = \sigma(w_0 X_0 + w_1 X_1 + b) = \frac{1}{1 + e^{-3/4}} \approx .6792$

$\text{output} = \frac{3}{4} + \frac{3}{4} - \frac{1}{2}(.6792) = \underline{0.5802}$