**Assignment two**

* Q1) Bagging is a technique which involves multiple bootstrap samples of the training data, where each sample is created by random selection of the observations with replacement from the original dataset. After which a separate model is trained on each sample and then the predictions from all of the models are aggregated to form a final prediction.

The main idea behind bagging is to reduce the variance of the model in which multiple models trained on different samples of the given data are combined together. Bagging is able to reduce the variance of a model by reducing the impact of individual observations that might have a larger influence on the single model. While Bagging may be effective for dealing with overfitting, it is not as effective as dealing with underfitting, because bagging only improves the variance of the model not the bias. Boosting may be more helpful when looking to address the bias.

* Q2) Bagging models are computationally more efficient compared to boosting models with the same number of weak learners because of how the two techniques are utilized.

In bagging, each model is trained independently on a bootstrap sample of the data. This means the training of each model can be parallelized which will allow for faster training and scalability to larger datasets. Whereas in boosting each model sequentially trained to correct the errors of the previous model.

This means that the training time increases with each additional weak learner, as the later models need to correct the errors of the previous models and ss a result, boosting can be computationally more expensive, especially when training a large number of weak learners. Overall, bagging is a simpler and more computationally effective technique than boosting, but boosting can sometimes achieve better performance in terms of accuracy.

Q3) Ensemble models work by combining the predictions of multiple models to create a more accurate final prediction. However, if the individual models are not accurate or diverse enough, then combining them may not improve the overall performance. If each decision tree model is not performing better than a random model and the models are very similar to each other, then it is unlikely that combining them into an ensemble model will significantly boost the performance.

In James’ case, it might be better to investigate why the individual decision tree models are not performing well. It is possible that the models are overfitting or underfitting the data, or that the features used for training the models are not informative enough.

Before James creates an ensemble model, it is important to ensure that the individual models are accurate and diverse enough to provide complementary predictions. Otherwise, the ensemble model may not be able to overcome the limitations of the individual models.

* Q4) In order to determine the information gain we should first calculate the entropy of the original dataset and the weighted entropy of the two resulting subsets. Finally, we calculate the information gain by subtracting the weighted entropies from the entropy of the original dataset:

See my formula below

**The entropy of the original dataset is**:

p(+): probability of an object being edible = 9/14

p(-): probability of an object being non-edible = 5/14

Entropy = - p(+) log2 p(+) - p(-) log2 p(-) = - (9/14) log2 (9/14) - (5/14) log2 (5/14) ≈ 0.940

**The weighted entropy of the two resulting subsets after splitting based on the "Size" attribute**.

Subset 1 (Size = Small):

p(+): probability of an object being edible = 5/7

p(-): probability of an object being non-edible = 2/7

Entropy = - p(+) log2 p(+) - p(-) log2 p(-) = - (5/7) log2 (5/7) - (2/7) log2 (2/7) ≈ 0.863

Subset 2 (Size = Large):

p(+): probability of an object being edible = 4/7

p(-): probability of an object being non-edible = 3/7

Entropy = - p(+) log2 p(+) - p(-) log2 p(-) = - (4/7) log2 (4/7) - (3/7) log2 (3/7) ≈ 0.985

**The information gain by subtracting the weighted entropies from the entropy of the original dataset:**

Information gain = Entropy - [ (Size = Small) \* Entropy(Small) + (Size = Large) \* Entropy(Large) ] = 0.940 - [ (7/14) \* 0.863 + (7/14) \* 0.985 ] ≈ 0.048

Therefore, the information gain for splitting the dataset based on the "Size" attribute is approximately **0.048**.

* Q5) The m parameter, also known as the maximum number of features available at each split, determines the number of candidate features that can be considered at each split point, which can affect the performance and efficiency of the model.

If the m parameter is set too small, the model may not be able to capture all the relevant information in the dataset which will lead to the model underfitting. Additionally, if the m parameter is set too small, the model may become overly reliant on a few important features, leading to limited diversity in the forest. Alternatively, if the m parameter is set too large the model may become too complex and overfit the training data. This is because the trees in the forest will have access to too many features, leading to a high degree of correlation among them. As a result, the random forest may have low bias, but high variance. Therefore, it is important to set the m parameter optimally to balance the bias-variance tradeoff and maximize the model's performance. This can be achieved through cross-validation as well as other optimization techniques.

**Part 2**

* QB1) Based on the output, the attribute used at the top of the tree for splitting is "Price", with a split value of 94.5
* QB2) Based on the decision tree model, the estimated Sales for the given input record is 9.586.
* QB3) Based on the output, the optimal value of mtry is 2, which gives the best performance for the random forest model.
* QB4) Based on the output, the best value of mtry for the random forest model is 3, with an RMSE of 2.364640. Note that the performance of the model for mtry values of 2 and 5 is also close to the best value.