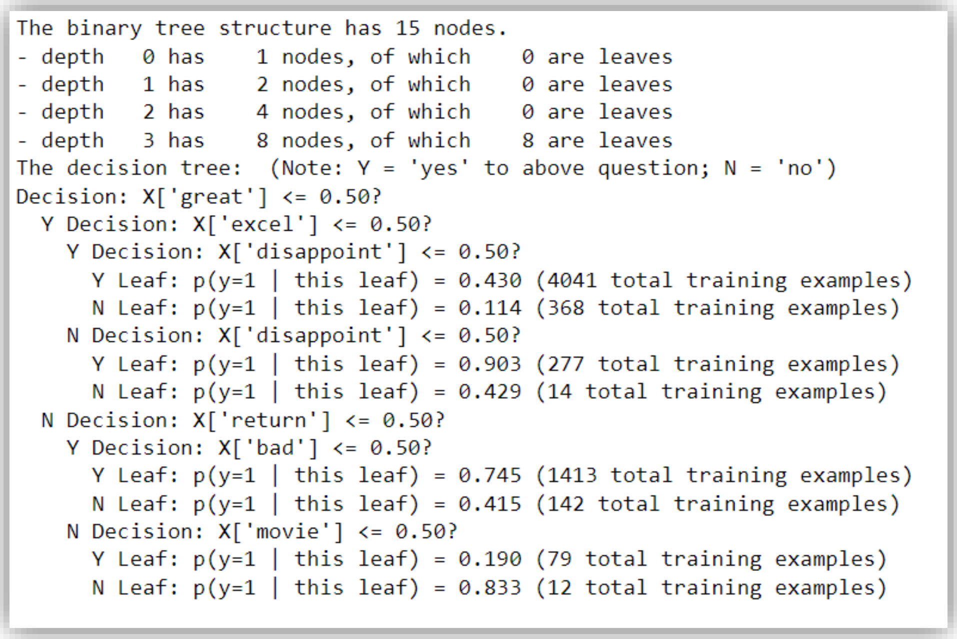
Problem 1: Decision Tree for Review Classification

Figure 1 in Report



```
The binary tree structure has 15 nodes.
- depth   0 has    1 nodes, of which    0 are leaves
- depth   1 has    2 nodes, of which    0 are leaves
- depth   2 has    4 nodes, of which    0 are leaves
- depth   3 has    8 nodes, of which    8 are leaves
The decision tree:  (Note: Y = 'yes' to above question; N = 'no')
Decision: X['great'] <= 0.50?
  Y Decision: X['excel'] <= 0.50?
    Y Decision: X['disappoint'] <= 0.50?
      Y Leaf: p(y=1 | this leaf) = 0.430 (4041 total training examples)
      N Leaf: p(y=1 | this leaf) = 0.114 (368 total training examples)
    N Decision: X['disappoint'] <= 0.50?
      Y Leaf: p(y=1 | this leaf) = 0.903 (277 total training examples)
      N Leaf: p(y=1 | this leaf) = 0.429 (14 total training examples)
  N Decision: X['return'] <= 0.50?
    Y Decision: X['bad'] <= 0.50?
      Y Leaf: p(y=1 | this leaf) = 0.745 (1413 total training examples)
      N Leaf: p(y=1 | this leaf) = 0.415 (142 total training examples)
    N Decision: X['movie'] <= 0.50?
      Y Leaf: p(y=1 | this leaf) = 0.190 (79 total training examples)
      N Leaf: p(y=1 | this leaf) = 0.833 (12 total training examples)
```

Short Answer 1a in Report

In Figure 1, the decision tree includes two internal node that test the existence of the word ‘disappoint’ in the reviews. For this internal node, both child leaf nodes fall under the same sentiment class, which is positive. (below 0.5 threshold) Each path in the decision tree essentially represents a unique combination of words/features. For example, in this tree, both child nodes represent two paths that are almost the same, but not fully because the left child node contains the feature ‘disappoint’ and the right doesn’t. Since the majority of the feature combination is the same, it makes sense that both leaf nodes predict positive for the sentiment class. However, the existence of the feature ‘disappoint’ affects the degree of the negative sentiment, and this is evident from the Gini Impurity. Lower Gini impurity means higher correct classification rate, and for this case, it is reasonable that left leaf node, which doesn’t have the word ‘disappoint’ in it is more likely to be positive.

Short Answer 1b in Report

- What is the max_depth of the best tree? 128
- What is the min_samples_leaf of the best tree? 1

Problem 2: Random Forests for Review Classification

Table 2 in Report

Important 10 Features

Feature	Importance
return	0.034
excel	0.030
great	0.029
worst	0.028
poor	0.027
disappoint	0.024
i_love	0.018
your_money	0.018
don't	0.018
bore	0.016

Unimportant 10 Features

Feature	Importance
you_want	1.670e-05
for_my	1.124e-05
hand	1.112e-05
comedi	6.725e-06
make_a	6.166e-06
but_i	4.991e-06
full_of	7.047e-07
month	6.479e-07
unless_you	5.020e-07
agree	1.092e-07

Short Answer 2a in Report

- What is the value of max_features of your best forest? 10
- What is the maximum value max_features could take for this dataset? 32
- Why do we want to tune max_features instead of setting it to its max possible value? Setting 'max_features' to its max possible value is not good practise for two important reasons. A smaller 'max_features' ensures that the trees in the forest are more diverse. Since each tree gets a different subset of features to consider at each split, the trees in the forest are less correlated. This improves the generalization of the forest because a diverse forest leverages the collective knowledge from trees that consider different perspectives of the data. Additionally, too high 'max_features' causes overfitting because considering every feature at every split can result in having too complex individual trees, The trees would perform well on the training data but fail to generalize on unseen data. Also, if there are features that are very strong predictors, there is less likelihood of the weaker but still informative features to be considered in the predictions.

Short Answer 2b in Report

In random forests, the hyperparameter 'n_estimators' represents the number of trees in the ensemble. The trade-off is between computation time and model performance. For higher 'n_estimators', the model performance is better because it averages the probability predictions from more trees. This reduces the variance and improves generalization. However, for too large 'n_estimators,' the computation time increases a lot because each tree in the ensemble needs to be trained. Unlike other hyperparameters, setting 'n_estimators' too large doesn't lead to overfitting. Each tree in a random forest is built on a random subset of data and makes its own predictions. Since each tree is somewhat independent of others, adding more trees does not cause the model to memorize the training data, instead it increases model generalization because the predictions are averaged.

Figure 3 and Caption in Report

Method	Max Depth	Num Trees	Train BAcc	Valid BAcc	Test BAcc
Simple Tree	**3**	**1**	**0.646**	**0.645**	**0.646**
Best Tree	**128**	**1**	**0.998**	**0.737**	**0.726**
Simple Random Forest	**3**	**100**	**0.816**	**0.797**	**0.773**
Best Random Forest	**32**	**100**	**0.972**	**0.847**	**0.816**

The best random forest performs the best on the data set with a balanced accuracy of 0.816. The overall ranking of the model performances in increasing order is simple tree, best tree, simple random forest, and best random forest. This agrees with the concepts learned in class for two major factors. First, random forest perform much better than simple tree because the averaging of the predictions of multiple trees make the resulting prediction more robust and reduces the variance.. Also, hyperparameter tuning finds the best set of hyperparameters, max depth and number of trees, that leads to more accurate models within the same model type. Thus, it makes sense that best tree comes at the third and best random forest at the first position for the best model performance metrics.

Problem 3: Analysis

Short Answer 3a in Report

The runtime complexity of making a prediction for a single test feature vector using a trained decision tree fore regression is O(D), where D is the depth of the tree. The tree evaluates a feature vector starting from the root travelling through a node in on each level of the tree based on the threshold at each node and reaches a leaf node at the end. This path from the root to a leaf node depends on the depth of the tree since the tree makes a comparison once at every level. Also, this comparison against the threshold value in each node is a constant time operation. Since the time complexity depends solely on the path length and the maximum length of a path from the root to a leaf is the depth of the tree, the time complexity of making a prediction equals to O(D). Neither the number of features F not the number of training examples N have a direct affect on this prediction time.