Homework 4 - Math 4803

Dennis Goldenberg

March 2023

1 Theoretical Questions

1. Consider the weakest link pruning method of regression trees. Suppose that the current value of α implies that we should prune a section S_t of the current tree to node t. For the sake of notation, suppose the original tree is T_0 and the pruned tree is T_1. Show that, after this pruning, all nodes that are upstream of node t (those nodes t' for which node t is in $S_{t'}$), satisfy $g_1(t') \geq g_0(t')$, with g defined in the lecture notes.
Solution

Proof. Consider a node t' upstream from node t. If it was decided that a t' should be pruned, then all of the downstream nodes would be pruned as well. Therefore, as t is downstream from t', we can deduce that t, and all of the nodes downstream from t would be pruned. From this, we can conclude:

- $|(S'_t)_1| \leq |(S'_t)_0|$ because, in T_1, S'_t consists only of those nodes downstream from t', but upstream from and including t, while $(S'_t)_0$ includes all such nodes and nodes downstream from t as well, as they have not been pruned yet.

Furthermore, in the formula described in the lecture notes, $Q_{t'}$ is the sum of mean squared errors over all data-points that traverse through t'. Note that in T_1, all of the data points that used to traverse through t down to leaves in T_0 now stop at t. Let us call this value $(Q_t)_1$. Any data points that traverse through t' but not t have their RSS unchanged; therefore the relationship between $(Q_{t'})_1$ and $(Q_{t'})_0$ is as follows:

$$(Q_{t'})_1 = (Q_{t'})_0 - \sum_{\ell \in S_t} Q_l(T_0) + (Q_t)_1$$

The subtraction of RSS of leaves in S_t is due to the fact that S_t has already been pruned and doesn't exist in T_1. Now, we examine the term in the formula that is $\sum_{\ell \in S_{t'}} Q_\ell(T)$. Note that, all leaves in the subsection S_t have been pruned already, so they need to be subtracted from this value in T_0 to account for this.

However, the t node itself is a leaf in T_1, so this needs to be added back (this is exactly the $(Q_t)_1$ value we defined earlier); therefore, we get the relationship:

$$\sum_{\ell \in S_{t'}} Q_\ell(T_1) = \sum_{\ell \in S_{t'}} Q_\ell(T_0) - \sum_{\ell \in S_t} Q_\ell(T_0) + (Q_t)_1$$

Therefore, if we put all of these results together:

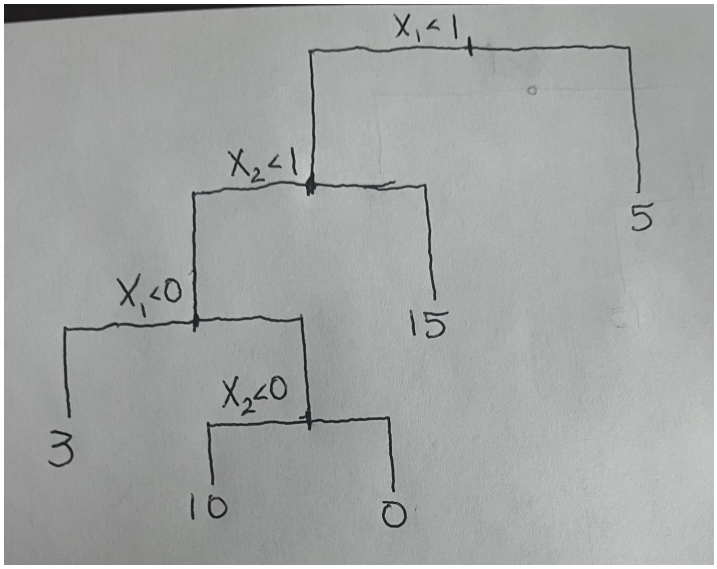
$$\begin{aligned}
g_1(t') &= \frac{(Q_{t'})_1 - \sum_{\ell \in S_{t'}} Q_\ell(T_1)}{|(S'_t)_1| - 1} \\
&= \frac{(Q_{t'})_0 - \sum_{\ell \in S_t} Q_l(T_0) + (Q_t)_1 - \left(\sum_{\ell \in S_{t'}} Q_\ell(T_0) - \sum_{\ell \in S_t} Q_\ell(T_0) + (Q_t)_1\right)}{|(S'_t)_1| - 1} \\
&= \frac{(Q_{t'})_0 - \sum_{\ell \in S_{t'}} Q_\ell(T_0)}{|(S'_t)_1| - 1} \\
&\geq \frac{(Q_{t'})_0 - \sum_{\ell \in S_{t'}} Q_\ell(T_0)}{|(S'_t)_0| - 1} \text{ (as } |(S'_t)_0| \leq |(S'_t)_1|) \\
&= g_0(t')
\end{aligned}$$

So, $g_1(t') \geq g_0(t')$. □

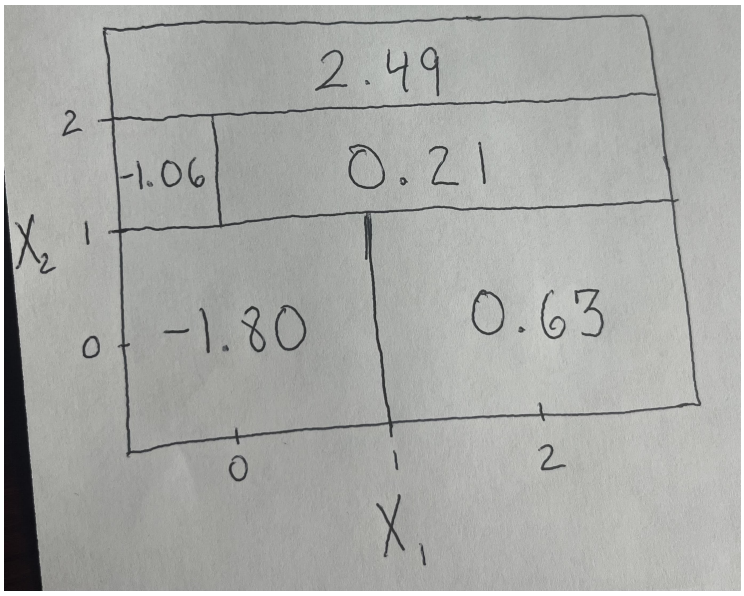
2. Section 8.4, Exercise 4

Solution

a) - The following tree corresponds to the partition in part a):



b) - The following partition corresponds to the tree in b):



3. Section 8.4, Exercise 5

Solution

The following are the estimates produces for the value $\mathbb{P}(\text{Class is red}|X)$:

$$0.1, 0.15, 0.2, 0.2, 0.55, 0.6, 0.6, 0.65, 0.7, \text{ and } 0.75$$

Since there are two classes, our threshold for a "yes" prediction will be ≥ 0.5. First, we evaluate using "majority voting". Note that, letting $N = 10$ representing the number of estimates:

$$\frac{\sum_{i=1}^{N} I_i\{\mathbb{P}_i(\text{class is Red}|X) \geq .5\}}{N} = \frac{6}{10} = .6 \geq .5$$

Therefore, more than half of predictions assign "red" to that specific value of X; thus, via **majority voting**, we assign the **red** class to X.
Next, we evaluate using average probability:

$$\begin{aligned}\frac{\sum_{i=1}^{N} \mathbb{P}_i(\text{Class is red}|X)}{N} &= \frac{.1 + .15 + .2 + .2 + .55 + .6 + .6 + .65 + .7 + .75}{10} \\ &= \frac{4.50}{10} = .450 < .5\end{aligned}$$

Thus, the average probability is below .5, meaning that we say no to assigning "red" to X. Since there is only one other class, "green", we must assign X this value, so, via **average probability**, we assign the **green** class to X.

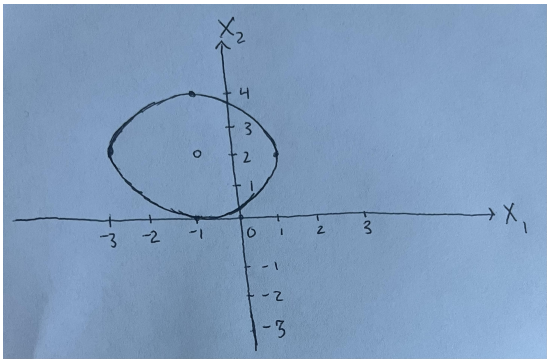
4. Section 9.7 Exercise 2

Solution

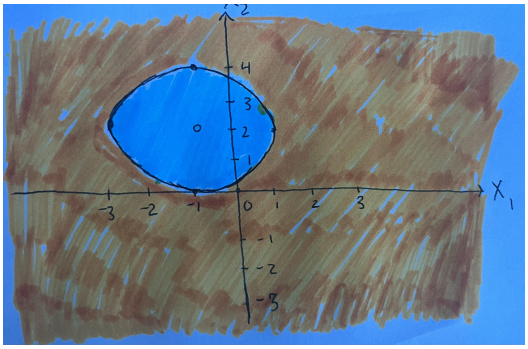
a) - Note that the variables in the equation can be shifted around:

$$\begin{aligned}
&(1+X_1)^2+(2-X_2)^2=4\\
&\Rightarrow (2-X_2)^2=4-(1+X_1)^2\\
&\Rightarrow 2-X_2=\pm\sqrt{4-(1+X_1)^2}\\
&\Rightarrow X_2=2\pm\sqrt{4-(1+X_1)^2}
\end{aligned}$$

Note that this is the equation of a circle, radius $\sqrt{4}=2$, centered at $(-1,2)$. The following is a rough sketch:



b) - This circle creates two distinct regions in the $\mathbb{R}^2$ plane. Note that 4 is just the representation of the radial distance squared from the center point $(-1,2)$. Thus, points closer to $(-1,2)$ have a radial distance less than 2, and points further have one more than 2; the following sketch was produced:



Here, we have the following legend:

$$\begin{aligned}
&\text{Light Blue (within circle): } (1+X_1)^2+(2-X_2)^2\leq 4\\
&\text{Orange (outside of circle): } (1+X_1)^2+(2-X_2)^2>4
\end{aligned}$$

Note that, as the curve represents the equation $(1 + X_1)^2 + (2 - X_2)^2 = 4$, all points on the circle are in the light blue region.
c) - In this case, the blue class corresponds to the orange region from b), and the red class corresponds to the complement, or the light blue region. We plug each point into the equation and solve:

$$\begin{aligned}
&\text{(0,0): } (1+0)^2 + (2-0)^2 = 1 + 4 = 5 > 4 \\
&\text{(-1,1): } (1+(-1))^2 + (2-1)^2 = 0 + 1 = 1 \leq 4 \\
&\text{(2,2): } (1+2)^2 + (2-2)^2 = 9 + 0 > 4 \\
&\text{(3,8): } (1+3)^2 + (2-8)^2 = 16 + 36 = 52 > 4
\end{aligned}$$

So, **(0,0), (2,2), and (3,8)** are classified as **blue** and **(-1, 1)** is classified as **red**.
d) - The curve represents $(1 + X_1)^2 + (2 - X_2)^2 = 4$. If we expand the left side of this equation, we get:

$$\begin{aligned}
(1 + X_1)^2 + (2 - X_2)^2 &= 1 + 2X_1 + X_1^2 + 4 - 4X_2 + X_2^2 \\
&= 5 + 2X_1 + X_1^2 - 4X_2 + X_2^2
\end{aligned}$$

Call this equation $f(X)$. Note that we can write this equation as:

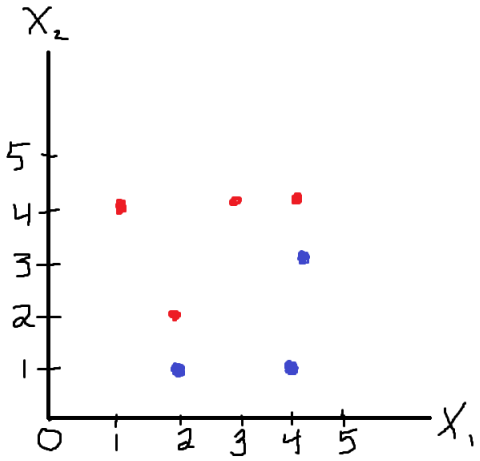
$$f(X) = \beta_0 + \beta_1 X_1 + \beta_2 X_1^2 + \beta_3 X_2 + \beta_4 X_2^2$$

Here, $\beta_0 = 5, \beta_1 = 2, \beta_2 = 1, \beta_3 = -4, \beta_4 = 1$. Thus, $f(X)$ is linear in parameters; a.k.a it is just a linear combination of the features X_1, X_1^2, X_2, and X_2^2. From the original problem, we can deduce that if $f(X) > 4$, we predict the label **blue** for X, but if $f(X)$ is ≤ 4, we predict the label **red**. since this is a constant, and $f(X)$ is linear in parameters, this is a linear decision boundary.

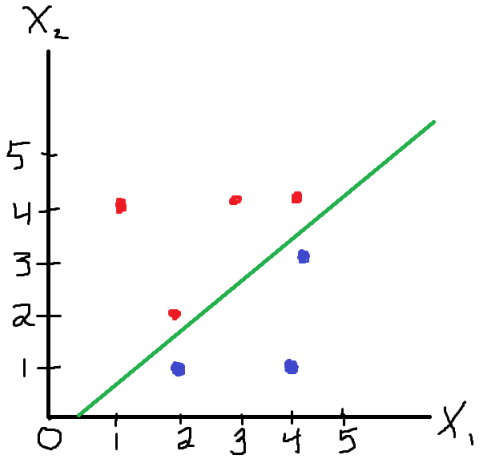
5. Section 9.7 Exercise 3

Solution

a) - The observations were color coded by their label (or Y value) and sketched here:



b) - The maximal margin classifier looks like this (line in green):



Note that the points of opposite classification closest to each other (and therefore the support points for the maximal margin classifier) are Observations 2,3,5, and 6 ((2,2), (4,4), (2,1), and (4,3) respectively). Note that (2,2) and (2,1) are 1 unit away from each other by Euclidean distance despite having different labels, and the same goes for (4,3) and (4,4). Thus, the maximal margin classifier has to go through the midpoint of each pair of points to minimize the sum of the euclidean distance from both. This happens to be (2, 1.5) and (4, 3.5) (as the support pairs have the same X_1 value, but X_2 values differing by 1). So our decision boundary for our maximal margin classifier has to satisfy the following system:

$$4\beta_1 + \beta_0 = 3.5$$
$$2\beta_1 + \beta_0 = 1.5$$

Subtracting the top of the equation from the bottom, we can arrive at our coefficients:

$$(4\beta_1 + \beta_0) - (2\beta_1 + \beta_0) = 3.5 - 1.5$$
$$\Rightarrow 2\beta_1 = 2 \rightarrow \beta_1 = 1$$
$$\Rightarrow 4(1) + \beta_0 = 3.5 \rightarrow \beta_0 = -.5$$

Therefore, our decision boundary hyperplane is as follows:

$$X_2 = X_1 - 0.5 \Rightarrow 0.5 - X_1 + X_2 = 0$$

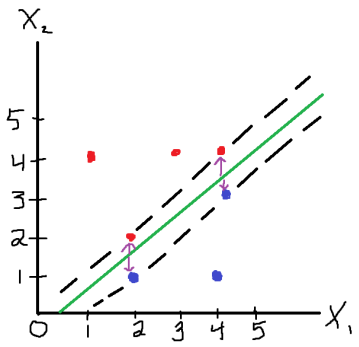
c) If we plug in the point (2,2), a red-labeled point, we get that:

$$0.5 - 2 + 2 = 0.5 > 0$$

So points "above" the maximal margin classifier are labelled red, as indicated in the sketch. Therefore the classifier rules are as follows:

- If $0.5 - X_1 + X_2 > 0$, label the point **red**.
- Otherwise (when $0.5 - X_1 + X_2 \leq 0$), label the point **blue**.

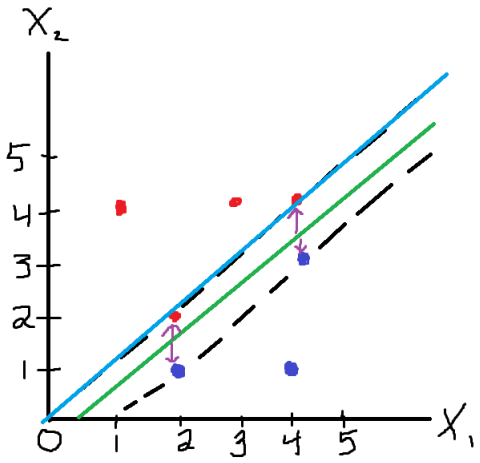
d) - The margin is colored purple and looks as follows:



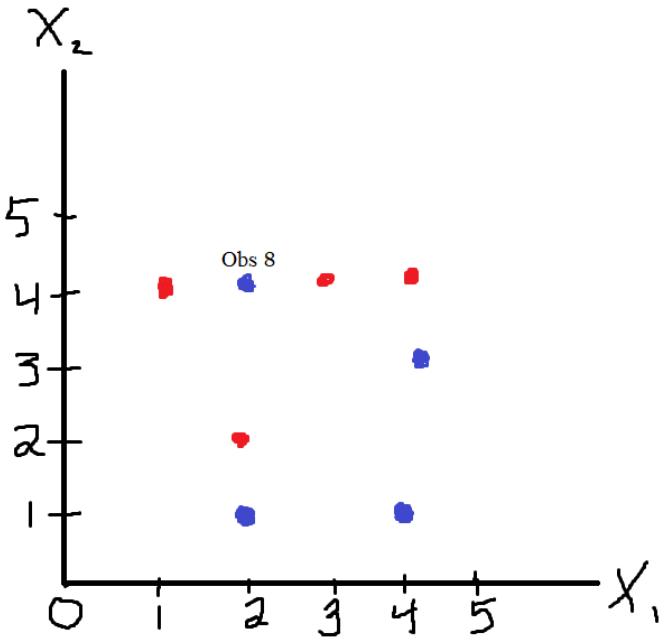
e) - The support vectors were already outlined in part b), and they are the points that the margin is drawn to, but, to restate it, the support vectors are the following points:

- Observation 2 or (2,2) - red
- Observation 5 or (2,1) - blue
- Observation 3 or (4, 4) - red
- Observation 4 or (4,3) - blue

f) - Note that Observation 7, or (4,1) is not a support vector; therefore, it is not one of the points creating the bounds for the potential hyper-planes that separate the classes perfectly, as it is it much farther away from points labelled red. If it were to be moved sightly (say to (3,1)), it still wouldn't be a support vector, as the closest opposite class point is still farther from it than Observation 5 (or (2,1)) and Observation 4 (or (4,3)); thus, it still wouldn't be a support vector, and not impact the maximal margin classifier at all.
g) - The non-optimal hyperplane is in turquoise in the following graph:



It passes through the points (2,2) and (4,4); since a hyperplane in 2 dimensions is a line, and we have these two points, we can determine the line to be $X_2 = X_1$, or $X_2 - X_1 = 0$. Not that this is still a separating plane; you just label "red" when $X_2 - X_1 \geq 0$ and "blue" otherwise.
h) - We can add Observation 8 like this:



Note that observation 8, a observation with a "blue label" is surrounded by red observations. There is no way to draw a line to separate observation 8 and the two red observations (one on either side), as no single line can effectively but Observation 8 on one "side" and those two red observations on the "other side".

2 Programming Questions

6. Extension of Section 8.4, Example 7

Solution

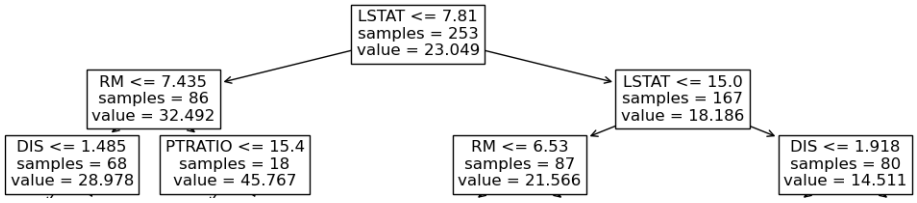
a) - Since regression trees are subject to high variance, one way to combat this is through pruning; specifically, the method I used was cost-complexity pruning (also known as weakest link pruning), trying to find the sub-tree T of some initially constructed tree T_0 that minimizes the following formula:

$$\sum_{m=1}^{|T|} \sum_{i:x_i \in R_m} (y_i - \hat{y}_{R_m})^2 + \alpha |T|$$

To accomplish this, I first created the pandas data frame from the Boston Housing dataset and used python's scikit-learn package to split the data into training and testing sets (50-50 split, as provided in the problem). My goal then was as follows: find the right α, and therefore the right sub-tree of T_0, that minimizes the test mean squared error. To do this, I have to effectively test all of the different trees according to the above formula. Luckily, scikit-learn's tree package has a method (titled "cost_complexity_pruning_path") that iteratively finds the smallest α values that correspond to an extra node pruning, ordered from least to greatest. Deleting the biggest α of this list (as that α corresponded to the a sub-tree that is just the root node), I fitted a tree using each α value generated using that method, and calculated each mean-squared error. I found the minimum mean-squared error, and the corresponding α value that generated it, shown below:

```
Optimal Alpha:  0.17456306144445171
MSE of Optimal Tree:  21.9259561096295
```

Then, I plotted the tree itself (the optimal tree had **23** nodes). The tree can be generated by running the code and expanding the screen; it is too small to show useful information here. However, zooming in on the first several layers, we can see the splits:



The features that were split on early are as follows:

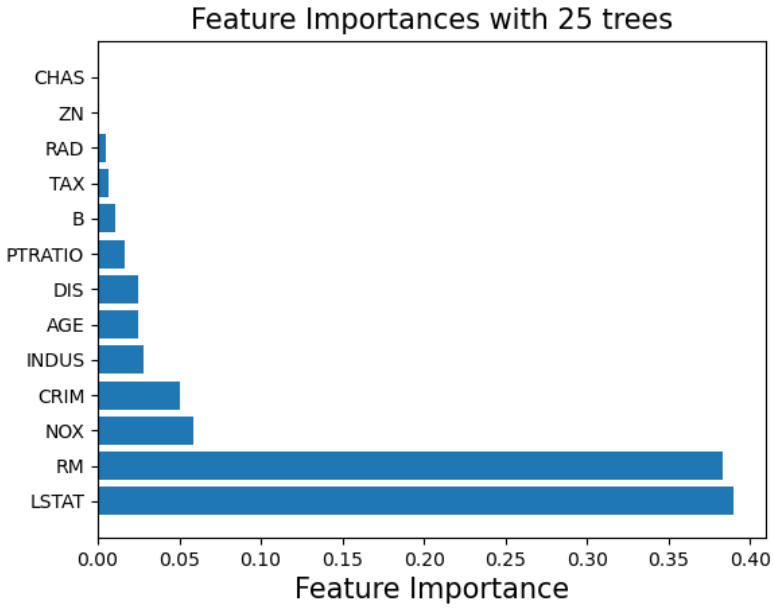
- LSTAT - This is the % of the population in that town that is lower-class. This has an intuitive correlation with housing prices, as this affects the prices people can afford.

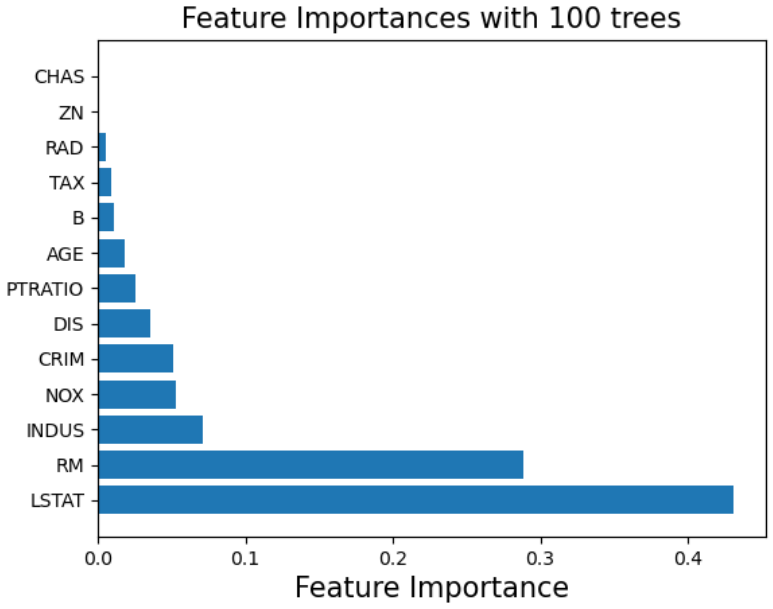
- RM - This value represents the average number of rooms per house in the town. The more rooms there are, the costlier houses are, so this variable makes sense as an important feature as well.
- DIS - This the weighted Distance to five Boston employment centers. This value likely has a strong correlation with the unemployment rate, which in turn affects purchasing power of people in the area, making this an important feature.
- PTRATIO - This corresponds to pupil-teacher ratio in the town. The smaller the ratio is, the less students per teacher, meaning education is likely of higher quality. Better educated people tend to make more money, and therefore have higher purchasing power, driving up the average price of homes.

b) - To create this random forest, similar steps were taken as in a) to set up the Boston Housing data set for training and testing. However, as cost complexity pruning is based on the actual original tree generated, and, as the random forest can only consider 6 of 13 possible features for each split, making each individual tree pretty uncorrelated with the others, I decided to forgo pruning for this case. Since there was no pruning, I made the smallest number of leaves in a node 5 to prevent added, mostly unneeded splits in any individual tree. The random forests were generated (first with 25 trees and then 100), and produced the following Mean Squared Error values when tested:

```
MSE with 25 trees:  15.6753873909545 96
MSE with 100 trees:  16.0918588147502 64
```

Note that the models performed roughly the same, with the 25 tree model slightly outperforming the higher-iteration 100 tree model. I also wanted to know which features were most important. Scikit-learn's random Forest regressor has an attribute called "feature_importance_", which gives the proportion of Residual Sum of Squares reduction that was caused by each feature. I plotted this for the 25 tree Random Forest, as well as the 100 tree random forest. The results are shown in the page below. Note that, in both cases, the two most important variables were LSTAT and RM, matching what our decision tree regressor seemed to suggest. However, after those 2, the similarity decerases; the random Forest tended to split more on NOX (nitric oxide concentration), INDUS (proportion of non-retail business acres), and CRIM (per capita crime rate per town), whereas the Decision Tree Regressor focused next on PTRATIO and DIS, with the inferred reasons discussed above. Nitric oxide concentration relates to quality of air and higher quality of air would make more desirable and thus higher priced houses. Houses in safer areas would also likely go for more; this is the impact crime rate would have. Finally, the more office buildings in an area, the likely higher prices are to be because commuting to work is shorter. So, NOX, CRIM, and INDUS each have an intuitive link, in my opinion more so than PTRATIO and DIS; thus, the results of the random forest seem to make slightly more intuitive sense.



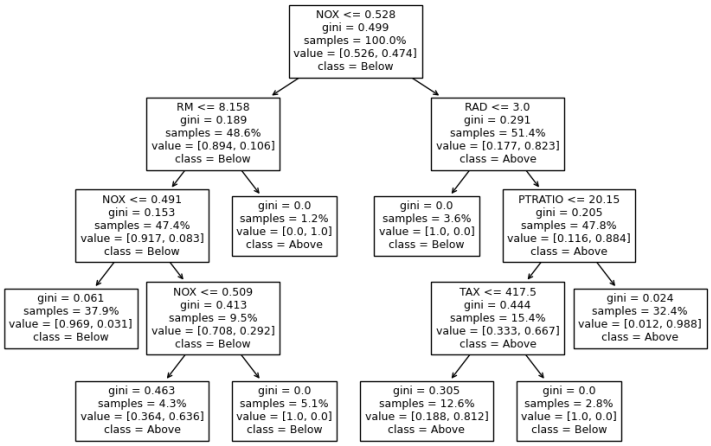


c) - This question is similar to part a, but a classification problem as opposed to a regression problem. The training and testing data is split in the same way, and the original tree is designed the same way, with the exception that I specified a

lower bound for the number of samples in a leaf node as 3, and the Gini coefficient was used to determine splits. Since this classification only had 2 distinct classes: "Above" the median and "Below" the median, there is less flexibility for the labels than in regression, so I wanted to avoid overfitting. Then, as in a), the "cost_complexity_pruning_path" method was used in order to generate all of the smallest alphas that correspond to iteratively pruned trees. Note that, in classification, test error is the proportion of incorrect classifications. Therefore, over all trees, the one with the lowest error rate had the following alpha value (with the corresponding error rate following):

```
Optimal Alpha:  0.011228889687387708
Error Rate of Optimal Tree:  0.05138339920948617
```

The tree that corresponds to this alpha and error rate is shown below:

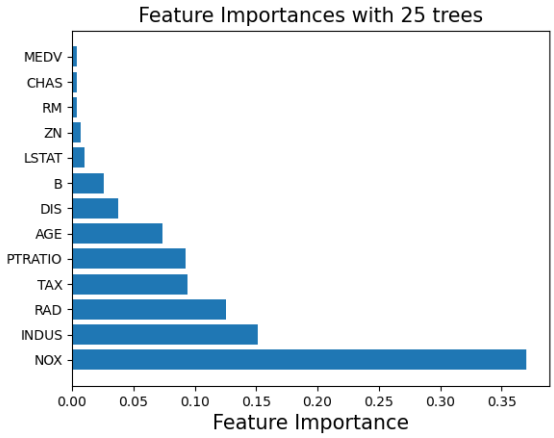


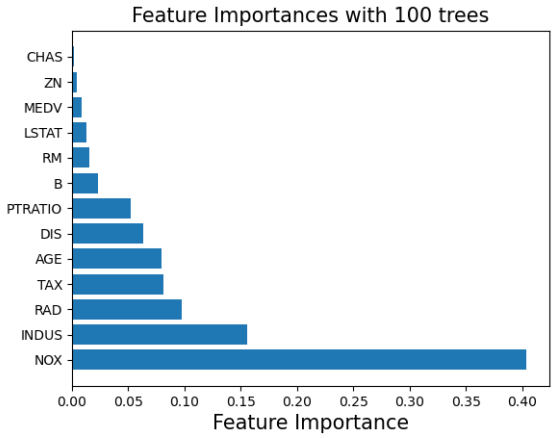
What's noteworthy here is that the value that was split on most often was NOX, or the parts per ten million of nitric oxide concentration. It seems as though air that is cleaner, or less filled with nitric oxide correlating with below-median crime rates in a town. Also important was the number of rooms per house on average in a town, or RM, with smaller numbers of rooms correlating with lower than median crime rates, and RAD, radial distance from highways, with greater distance from highways correlating with higher than median crime rates. My intuition on this question is not as good as parts a) and b), but all of these conclusions seem relatively reasonable; cleaner air leads to more stable people and criminals typically prefer to burgle from bigger homes, but the results from radial highway distance are a bit puzzling.

d) - The setup for this question is actually very similar to b). No cost complexity pruning was used for any of the trees in the random forest, and, also similar to b), I made the smallest number of leaves in a node 5 to prevent unneeded splits in any individual tree, and used Gini coefficient methodology to determine splits. The random forests were generated, and produced the following Error rate when tested:

```
Error Rate with 25 trees:  0.055335968379446 64
Error Rate with 100 trees:  0.047430830039525 69
```

The models performed roughly the same, with the 100 tree forest slightly edging out the 25 tree forest in terms of accuracy; both are similar to the performance of the Decision Tree classifier with the optimal α value. As with b), it is also interesting to examine the importance of the features split on. The feature importance values were generated and can be seen below:





In both cases, it is clear that the most important feature split on to reduce errors in classification was NOX, which aligns with our results in the decision tree classifier. RAD, or radial distance from highways, was also important, and this also aligns with our decision tree classifier. What's notable is that, in both random forests, it seems as though INDUS, or proportion of non-retail business acres in town was the 2nd most important feature. This intuitively makes sense, as those areas with high INDUS values tend to be more urban areas with many office buildings, and these urban areas tend towards higher crime rates.

7. Chapter 9, Exercise 9.5 (From *Elements of Statistical Learning*, skip part a) and d)

b) - Done in `hw4_coding.py`

c) - The trees were generated with the specified number of nodes (1, 5, and 10) and the response variable was generated 10 different times for 10 different simulations to get a better estimate for degrees of freedom, as outlined in the problem. Note that their formula for degrees of freedom is:

$$df = \frac{\sum_i \text{cov}(y_i, \hat{y}_i)}{\sigma^2}$$

Since all $\sigma^2 = 1$, this formula simplifies to:

$$\begin{aligned} df &= \frac{\sum_i \text{cov}(y_i, \hat{y}_i)}{1} \\ &= \sum_i \frac{(y_i - \bar{y})(\hat{y}_i - \bar{\hat{y}})}{1} \\ &= \sum_i (y_i - \bar{y})(\hat{y}_i - \bar{\hat{y}}) \end{aligned}$$

We sum this df calculation for each different tree over the 100 different data points, and the following degrees of freedom estimates were generated:

```
Degrees of Freedom, 1 Leaf:  1.0225545937048397
Degrees of Freedom, 5 Leaves:  30.76466585158326
Degrees of Freedom, 10 Leaves:  53.80206499662898
```

So, the degrees of freedom for a tree that is just the root node is ~ 1, the 5-leaf tree has about $30 - 31$ degrees of freedom, and the 10 leaf tree has about $53 - 54$.

e) Note that, for any $\hat{y}$ values that fall into the same terminal leaf node k, their value is the same: $\bar{y}_k$, or the average of the actual y values in said node. So, for a given y_i that is located in terminal node k_i, how can we make sure $\hat{y}_i = \bar{y}_{k_i}$. First, we must know how many y values are in a given node k. For simplicity's sake, given m possible terminal nodes, we will assume that the y values are uniformly distributed; that is, each terminal node has $\approx \frac{n}{m}$ y values contained within it. To make sure we get that equality, we can construct the transformation matrix $\mathbf{S}$ as follows:

- For a given row i in $\mathbf{S}$ (which, when multiplied with y, gives $\hat{y}_i$)

$$s_{ij} = \begin{cases} \frac{1}{\frac{n}{m}} = \frac{m}{n} & \text{if } y_i, y_j \text{ in same terminal node} \\ 0 & \text{otherwise} \end{cases}$$

  (for the purposes of linearity, we are assuming that y_i and $\hat{y}_i$ are always in the same node to make the linearity work)

Since there are only $\frac{n}{m}$ nodes in k_i, those values would be averaged together in the following manner:

$$\hat{y}_i = \sum_{j=1}^{n} s_{ij} y_j = \frac{m}{n} \sum_{y \in k_i} y = \bar{y}_{k_i}$$

So, the linear transformation creates the right values. Since we by the identity property know that each y_i is in the same node as itself, we can deduce that:

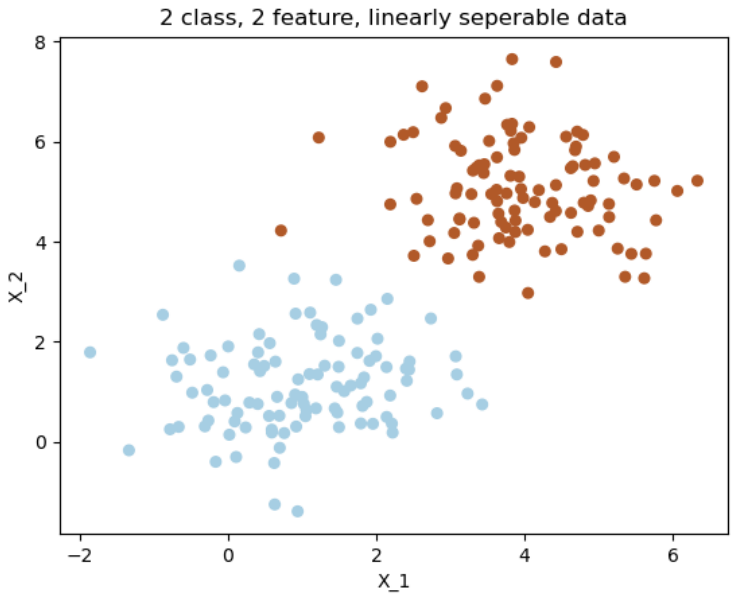
$$\text{Estimated Degrees of Freedom} = tr(\mathbf{S}) = \sum_{i=1}^{n} s_{ii} = n * \frac{m}{n} = m$$

This is an estimation significantly lower than my predicted values; however, this approximation makes intuitive sense, as there are m different possible values for $\hat{y}$. However, it is likely that the y values are not uniformly distributed among the nodes in my trees.

8. Section 9.7 Exercise 6

Solution

a) - This part was done in `hw4_coding.py`. The following graph was produced from 200 generated datapoints:



It is linearly separable because, when a Support Vector classifier with effectively no regularization (cost = 10000, so this classifier became a maximal margin classifier) produced a 0.0 error rate when predicting; thus, the data is linearly separable.

b) - The Cross-Validation that was performed was 5-fold validation. The following *cost* parameter values were chosen, along with the corresponding training error and testing error:

```
Cost parameters:  [0.0001, 0.0003, 0.0005, 0.0007, 0.0009, 0.001, 0.003, 0.005, 0.007, 0.009]
CV Train Errors:  [0.29625, 0.29125, 0.03125, 0.00375, 0.005, 0.005, 0.005, 0.00375, 0.00375, 0.00125]
CV Test Errors:  [0.325, 0.32, 0.02, 0.0, 0.005, 0.005, 0.005, 0.005, 0.005, 0.005]
```

Note that the trend in train Error mimics the test error almost perfectly. When the training error goes down, the test error goes down (this is the general trend as the C parameter increased, as regularization was weaker); similarly, when training error goes up, testing error also goes up.

c) - A 50-50 train-test split was performed, and an error rate generated for predictions done for each cost parameter. The following test errors were generated:

```
Testing Error:  [0.53, 0.53, 0.53, 0.3, 0.07, 0.03, 0.0, 0.0, 0.0, 0.0, 0.0]
```

The train test-split methodology had lowest error rates with the higher cost parameters (0.001, 0.003, 0.005, 0.007, 0.009) while the minimum test error for

the Cross-Validation was one of the smaller cost parameters (0.0007).
d) - The 50-50 train test split methodology likely preferred higher cost parameters (and therefore lower regularization) due to the nature of the data generation being Gaussian and not linear. Since it was likely that the training and testing data points had very similar layouts, the classifier that was produced at lower regularization levels was likely a very close approximation of the maximal margin classifier. The Cross-Validation approach, having 5 "folds", may have had more varied spread of data between folds; thus, less regularization actually led to some over-fitting. However, the intended outcome was to show more drastically the over-fitting to support vector points; in hindsight, creating the classifier itself and then generating random data in each hyperplane likely would have illustrated the point better.