Stats 315B: Homework 1

Joe Higgins, Austin Wang, Jessica Wetstone

Due 5/6/2018

#Question 1 ##Data Mining Marketing

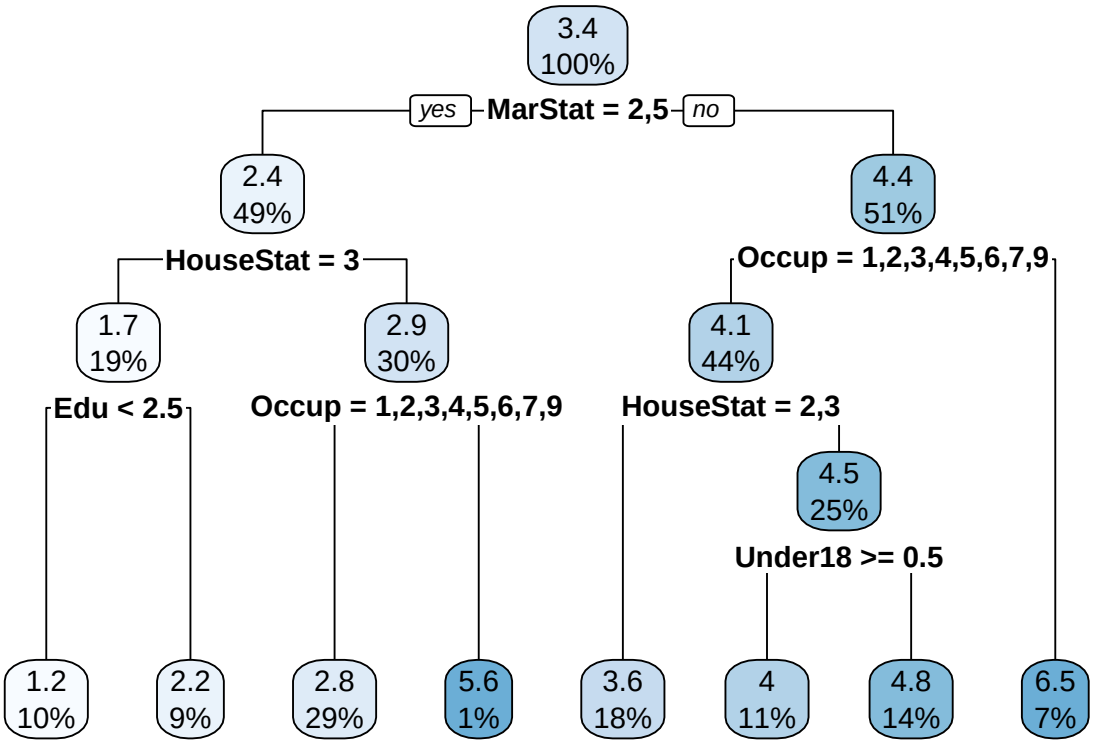
The data set age_stats315B.csv represents an extract from a commercial marketing database. The goal is to fit a regression tree to predict the age of a person from 13 demographic attributes and interpret the results. Note that some of the variables are categorical: be sure to mark them as such using the R function as.factor, before running rpart. Use the RPART implementation of the decision tree algorithm to fulfill this task. Write a short report about the relation between the age and the other demographic predictors as obtained from the RPART output and answer the following questions:

The first primary split of the tree to predict age was on Marital Status. It split people into two groups of roughly equal size (49% and 51% of the total) comprised of mostly younger folks in one group, and mostly older in the other.

The younger side of the first primary split belonged to set {2. Living together, not married 5. Single, never married}. Intuitively, this makes sense and gels with our outside knowledge: people who have never been married tend to be younger and vice versa. Within the younger group of the primary split, the next split was on householder status, specifically whether or not a person was in the set {3. Live with Parents/Family}. Those who lived with their parents were younger, and those who owned or rented were older within this younger primary split. Those who lived with their parents and had yet to graduate high school were the youngest 10% of our tree. There is also a small terminal node of (1% of the total) the oldest subset of those never married: these people were never married → don't live with their parents → are retired.

The older side of the first primary split belonged to the set {1. Married, 3. Divorced or separated, 4. Widowed}. The next split within the older group defines oldest leaf in our tree, those who are retired. For those that are in the older primary split node but not retired, there are two other variables that further predict our age groups: home ownership, and whether or not there is at least one person in the house under 18. Overall, the splits gel with common sense understanding of age: marriage, home ownership, education level and retirement status all intuitively are good indicators of a person's age.

```
rm(list = ls())
data_path <- paste(getwd(),'/data',sep='')
setwd(data_path)

#Read and type data
age_data <- read.csv('age_stats315B.csv')
factor_columns <- c(
  'Occup',
  'TypeHome',
  'sex',
  'MarStat',
  'DualInc',
  'HouseStat',
  'Ethnic',
  'Lang'
)
age_data[factor_columns] <- lapply(age_data[factor_columns], as.factor)
fit <- rpart(age ~ ., data = age_data)
rpart.plot(fit)
```



(a) Were surrogate splits used in the construction of the optimal tree you obtained? What does a surrogate split mean? Give an example of a surrogate split from your optimal decision tree. Which variable is the split on? Which variable(s) is the surrogate split on?

Yes there were surrogate splits used. You can see them by calling `summary()` on the model. Surrogates are variables used to classify data points that have missing values for a given primary split feature. They are used to predict the primary split feature. One example in our data set is that the top surrogate for a split on Householder status is whether or not the example has no people under 18 in the household. Intuitively, this makes sense. The surrrogate of the surrogate is Education.

```
Primary splits:
HouseStat splits as RRL, improve=0.2932322, (92 missing)
Edu < 2.5 to the left, improve=0.2862099, (38 missing)
Occup splits as RLRRRLRRL, improve=0.2470701, (0 missing)
Under18 < 0.5 to the right, improve=0.2206589, (0 missing)
Persons < 2.5 to the right, improve=0.2131353, (189 missing)
Surrogate splits:
Under18 < 0.5 to the right, agree=0.776, adj=0.440, (92 split)
Edu < 2.5 to the left, agree=0.758, adj=0.396, (0 split)
Occup splits as RRRRRLRRL, agree=0.755, adj=0.387, (0 split)
Persons < 2.5 to the right, agree=0.741, adj=0.352, (0 split)
TypeHome splits as LRRRR, agree=0.732, adj=0.330, (0 split)
```

```
#summary(fit)
```

(b) Using your optimal decision tree, predict your age.

```r
#b)
ncols <- dim(age_data)[2]
joe_data <- data.frame(matrix(ncol = ncols, nrow = 0))
                              #1,2,3,4,5,6,7,8,9,10,11,12,13,14
joe_data <- rbind(joe_data, c(3,1,3,1,5,6,9,3,1, 4, 0, 2, 8, 1))
colnames(joe_data) <- colnames(age_data)
joe_data[factor_columns] <- lapply(joe_data[factor_columns], factor)
predict(fit, joe_data)
```

```
##        1
## 2.808754
```

Result: 2.8, on the upper end of 18 through 24. I'm 29. Gotcha tree!

#Question 2 ##Multi-Class Classification: Marketing Data.

The data set housetype_stats315B.csv comes from the same marketing database that was used for problem 1. Refer to the documentation housetype_stats315B.txt for attributes names and order. From the original pool of 9409 questionnaires, those with non-missing answers to the question "What is your type of home?" were selected. There are 9013 such questionnaires. The goal in this problem is to construct a classification tree to predict the type of home from the other 13 demographics attributes. Give an estimate of the misclassification error of an optimal tree. Plot the optimal tree if possible (otherwise plot a smaller tree) and interpret the results.

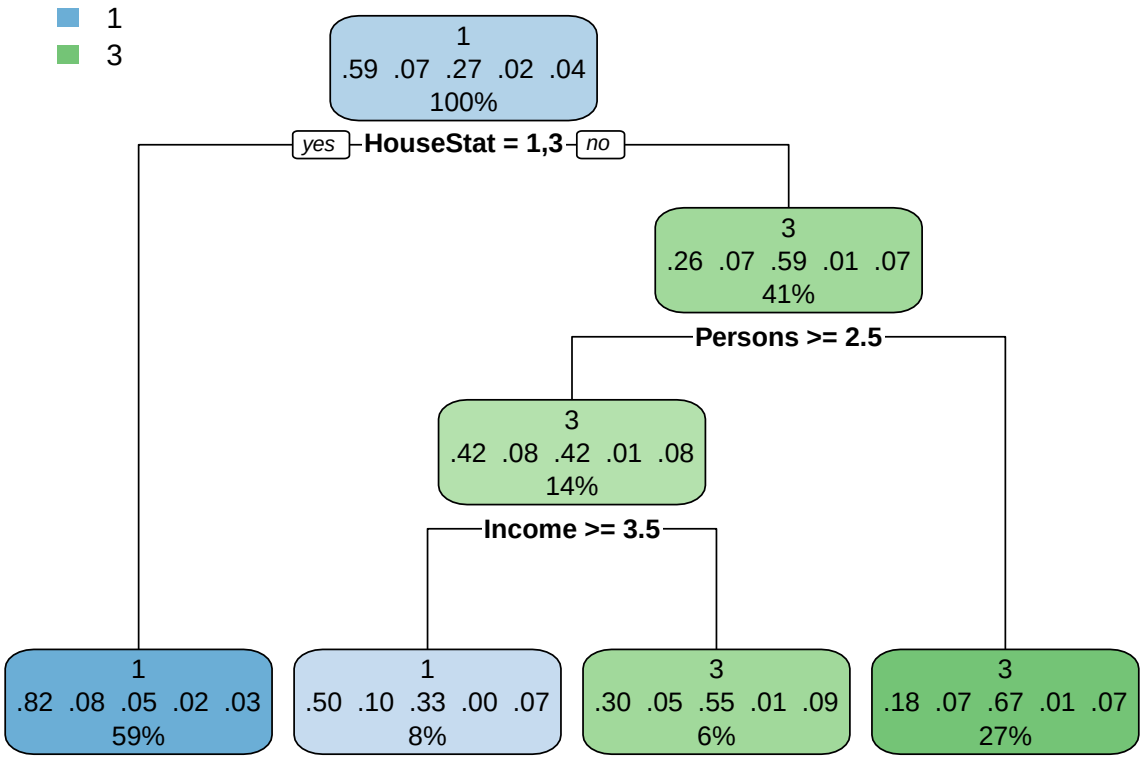
The misclassification rate is approximately 26.1%.

The first primary split of the tree divides our population into 2 groups: those that rent, and those that either own or live with their parents. Those that own or live with their parents are a large group, 59% of the population, and the tree predicts these examples live in houses (class 1). This makes intuitive sense.

The other half of the first primary split, renters, has two other variables it splits on to predict houses vs. apartments: having fewer than 3 people living in a household was indicative of apartments. If there were more than 3 people in the household, then relatively lower income was indicative of an apartment.

The terminal nodes in this example can only classify 2 out of the 5 potential classes: houses (class 1) and apartments (class 3). These two classes make up the vast majority of the examples in the data (see barplot), and there were no terminal nodes that had a majority vote for any of the 3 smallest classes.

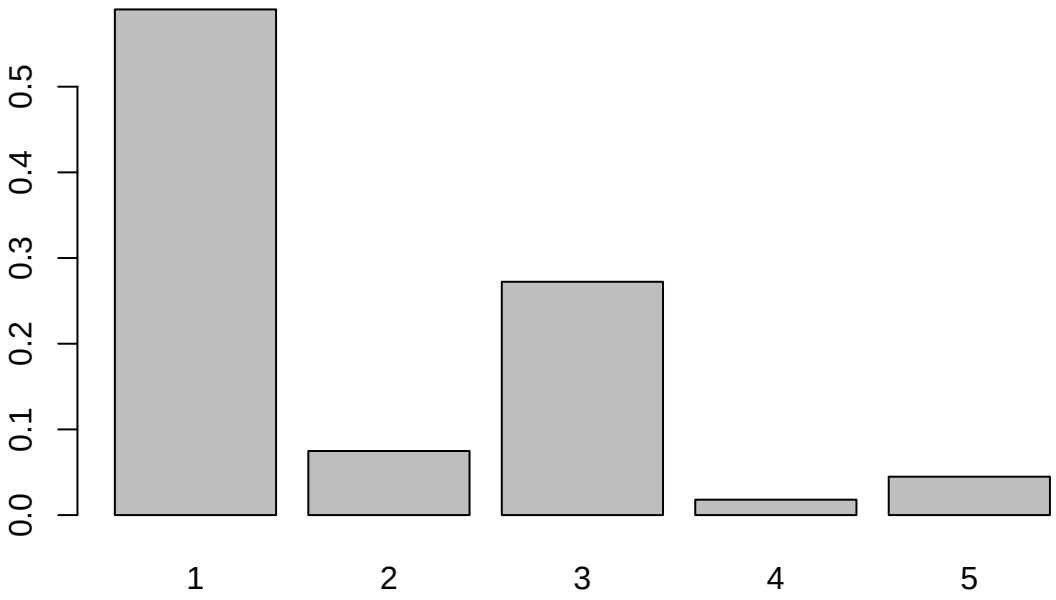
```r
rm(list = ls())
data_path <- paste(getwd(),'/data',sep='')
setwd(data_path)

#Read and type data
house_data <- read.csv('housetype_stats315B.csv')
factor_columns <- c(
  'TypeHome',
  'sex',
  'MarStat',
  'Occup',
  'LiveBA',
  'DualInc',
  'HouseStat',
  'Ethnic',
  'Lang'
)
house_data[factor_columns] <- lapply(house_data[factor_columns], as.factor)
fit <- rpart(TypeHome ~ ., data = house_data, method = 'class')
rpart.plot(fit)
```




```
#misclassification
y_hat <- predict(fit, house_data)
y_hat_max_p <- apply(y_hat,1,which.max)
y <- house_data$TypeHome
correct <- y == y_hat_max_p
pct_correct <- sum(correct)/length(correct)
1-pct_correct
```

```
## [1] 0.2614002
```

```
barplot(prop.table(table(house_data['TypeHome'])))
```


0.0
0.1
0.2
0.3
0.4
0.5
1
2
3
4
5

#Question 3 ###What are the two main reasons why a model that accurately describes the data used to build it, may not do a good job describing future data?

1. Any supervised learning model assumes stationarity in the data; namely that new examples will be drawn from the same $p(y|x)$ distribution as the training data. If the $p(y|x)$ of future data is different than that of the training data, then no matter how good our model approximated the old distribution, we have no guarantees for future data.
2. Overfitting the training data: Another reason that a model might not accurately describe future data is if it has a lot of variance; i.e. it has learned too much of the noise in the training data and thus will not generalize well.

#Question 4 ###Why can't the prediction function be chosen from the class of all possible functions?

If we expanded our function class to the class of all possible functions, then we could run into the following issues:

- The class of all possible functions doesn't have a feasible search strategy.
- There are many (infinite) solutions that minimize empirical risk so we will almost certainly overfit. While the class of all functions would obviously include the true target function, in searching for this function, we would have a high probability of encountering other functions that perfectly explained our training data. Having minimized the empirical prediction risk over our training data, we would stop searching before finding the target function. Our model would likely have zero bias but very high variance.

#Question 5 ###What is the definition of the target function for a given problem? Is it always an accurate function for prediction? Why/why not?

The target function F^* is the function that minimizes the expected loss over future predictions: $F^* = argmin_F E_{xy} L(y, F(x))$, where E_{xy} is the expected value of the joint distribution of $\{y, x\}$ and the loss function $L(y, F(x))$ is a measure of the cost to the user when $y \neq F(x)$.

The target function is not always an accurate function for prediction. It is possible that the set of features $\bar{x}$ is just not very predictive of y overall, e.g. y might actually depend on another set of latent variables that we didn't observe. It's also possible that y depends so much on random noise that it is impossible to predict y accurately (For example, suppose y was random noise).

#Question 6 ###Is the empirical risk evaluated on the training data always the best surrogate for the actual (population) prediction risk? Why/why not? In what settings would it be expected to be good?

The empirical risk evaluated on the training data is not always the best surrogate for the actual population prediction risk. An analogy to better help understand this is to treat empirical risk as the sample mean and the population risk as the true population mean. While the sample mean converges to the population mean by the law of large numbers, for small sample sizes the variance of the sample mean will be very large and thus could be far off from the true population mean. Similarly, in small samples, the empirical risk has a higher variance and thus may not estimate the population prediction risk well. However, in settings where our training data set is large, the variance will be much smaller and therefore we expect the empirical risk to be a good estimate.

In settings where we have a small dataset, the empirical risk may not be the best surrogate for the population prediction risk. In that case, using regularization techniques like enforcing an L1 or L2 penalty on the parameters in our model could lead to a better estimate for the population prediction risk.

#Question 7 ###Suppose the loss for an incorrect classification prediction is the same regardless of either the predicted value c_k or the true value c_l of the outcome y. Show that in this case misclassification risk reduces to the classification error rate. What is the Bayes rule for this case in terms of the probabilities of y realizing each of its values $\{Pr(y = c_k)\}_{k=1}^K$? Derive this rule from the general (unequal loss) Bayes rule, for this particular loss structure $L_{kl} = 1(k \neq l)$.

The misclassification risk is given by $r = E_{y\bar{x}}L(y, \hat{y}(\bar{x}))$. If the cost of misclassification is constant for all values k, l with $c_k \neq c_l$, then $L(y = c_l, \hat{y}(\bar{x}) = c_k) = L_{lk} = I(l \neq k) \implies L(y, \hat{y}(\bar{x})) = I(y \neq \hat{y}(\bar{x}))$.

Thus, $r = E_{y\bar{x}}I(y \neq \hat{y}(\bar{x}))$, which is simply the expectation of a Bernoulli random variable with parameter $p = P(y \neq \hat{y}(\bar{x}))$. Since the expectation of a Bernoulli random variable with parameter p is p, we find that:

$$\begin{aligned} r &= E_{y\bar{x}}I(y \neq \hat{y}(\bar{x})) \\ &= P(y \neq \hat{y}(\bar{x})) \\ &= \text{Classification Error Rate} \end{aligned}$$

We know that the general Bayes' optimal prediction rule states:

$$k^*(x) = argmin_{1 \leq k \leq K} \sum_{l=1}^{K} L_{lk} Pr(y = c_l|\bar{x})$$

Substituting in the particular loss structure $L_{kl} = 1(k \neq l)$,

$$\begin{aligned} k^*(x) &= argmin_{1 \leq k \leq K} \sum_{l=1}^{K} I(l \neq k) Pr(y = c_l|\bar{x}) \\ &= argmin_{1 \leq k \leq K} \sum_{l \neq k} Pr(y = c_l|\bar{x}) \end{aligned}$$

We have defined our classes such that each observation $\bar{x}$ must belong to exactly one class. Thus, we know $\sum_{l=1}^{K} Pr(y = c_l|\bar{x}) = 1$. So

$$\begin{aligned} k^*(x) &= argmin_{1 \leq k \leq K} \sum_{l \neq k} Pr(y = c_l|\bar{x}) \\ &= argmin_{1 \leq k \leq K} \{1 - Pr(y = c_k|\bar{x})\} \\ &= argmax_{1 \leq k \leq K} Pr(y = c_k|\bar{x}) \qquad \blacksquare \end{aligned}$$

#Question 8 ###Does a low error rate using a classification rule derived by substituting probability estimates $\{\hat{Pr}(y = c_k)\}_{k=1}^{K}$ in place of the true probabilities $\{Pr(y = c_k)\}_{k=1}^{K}$ in the Bayes rule imply accurate estimates of those probabilities? Why?

Not necessarily. Imagine a scenario in which we have a data set with large class imbalance (majority of examples are class 0, few examples are class 1). Then we can use a substitute probability derived from the proportions present in our training set (For example, $P(y = 0) = .98$ and $P(y = 1) = 0.02$) and achieve high accuracy. However, this will not generalize if the true population classes are more balanced (close to 50/50). In other words, this does not work well if our sample data is not representative of the population.

#Question 9 ###Explain the bias-variance trade-off.

The bias-variance trade-off states that in general if we increase model complexity, we tend to reduce squared bias but increase variance. Similarly, reducing complexity tends to increase bias and decrease variance.

An example of a high-bias low-variance estimating function is a constant (for a target function that is not a constant): there is a consistent and predictable bias, and there is no variance.

However, we can reduce bias by introducing more model complexity. In this specific example, we can change the estimating function to actually rely on the data, for instance. In general, as we increase model complexity, we increase the ability of our model to overfit to the noise in our training data. This leads to a higher variance model (One that won't generalize as well).

#Question 10 ###Why not choose surrogate splits to best predict the outcome variable y, rather than the primary split?

In the CART algorithm, surrogate splits $\{j(l, m), s(l, m)\}_{l=2}^{k}$ are chosen to be those splits that best predict the primary split $\{j(1, m), s(1, m)\}$.

If instead we chose a surrogate split to predict the outcome variable y, we may encounter a problem: the rest of the tree would no longer be optimal, because it is expecting observations of a certain type - those dictated by the primary split. Each of the subsequent splits were chosen to optimize (greedily) for data of a certain structure, and using a surrogate split that best predicts y instead of the primary split may send observations of an unexpected structure to its daughters.

#Question 11 ###Show that the values of c_m that minimize the squared-error risk score criterion are given by:

$$\hat{c}_m = \frac{\sum_{i=1}^{N} y_i I(x_i \in R_m)}{\sum_{i=1}^{N} I(x_i \in R_m)}$$

We will show this by taking the partial derivative of the score function $S(\mathbf{c}) = \sum_{i=1}^{N}[y_i - \sum_{k=1}^{M} c_k I(x_i \in R_k)]^2$ with respect to an arbitrary c_m:

$$\begin{aligned}
S(\mathbf{c}) &= \sum_{i=1}^{N}[y_i - \sum_{k=1}^{M} c_k I(x_i \in R_k)]^2 \\
\implies \frac{\partial S}{\partial c_m} &= 2\sum_{i=1}^{N}[y_i - \sum_{k=1}^{M} c_k I(x_i \in R_k)](-I(x_i \in R_m)) \\
&= -2\sum_{i=1}^{N} y_i I(x_i \in R_m) + 2\sum_{i=1}^{N}\sum_{k=1}^{M} c_k I(x_i \in R_k)(I(x_i \in R_m)) \\
&= -2\sum_{i=1}^{N} y_i I(x_i \in R_m) + 2\sum_{i=1}^{N} c_m I(x_i \in R_m)
\end{aligned}$$

Let $\hat{c}_m$ be the value that sets $\frac{\partial S}{\partial c_m} = 0$. Then:

$$\begin{aligned}
-2\sum_{i=1}^{N} y_i I(x_i \in R_m) + 2\sum_{i=1}^{N} \hat{c}_m I(x_i \in R_m) &= 0 \\
\implies \sum_{i=1}^{N} \hat{c}_m I(x_i \in R_m) &= \sum_{i=1}^{N} y_i I(x_i \in R_m) \\
\implies \hat{c}_m \sum_{i=1}^{N} I(x_i \in R_m) &= \sum_{i=1}^{N} y_i I(x_i \in R_m) \\
\implies \hat{c}_m &= \frac{\sum_{i=1}^{N} y_i I(x_i \in R_m)}{\sum_{i=1}^{N} I(x_i \in R_m)}
\end{aligned}$$

Finally, we show that this value is in fact the unique minimizer by examining the second derivative with respect to c_m:

$$\begin{aligned}
\frac{\partial S}{\partial c_m} &= -2\sum_{i=1}^{N} y_i I(x_i \in R_m) + 2\sum_{i=1}^{N} c_m I(x_i \in R_m) \\
\implies \frac{\partial^2 S}{\partial c_m^2} &= 2\sum_{i=1}^{N} I(x_i \in R_m) \\
&> 0
\end{aligned}$$

where the final inequality comes from the fact that at least one training example must exist in R_m for it to be defined.

Therefore, the second partial derivative with respect to c_m is strictly positive, so the score function is strictly convex in c_m and admits a unique minimum given by $\hat{c}_m = \frac{\sum_{i=1}^{N} y_i I(x_i \in R_m)}{\sum_{i=1}^{N} I(x_i \in R_m)}$ ■.

#Question 12 ###Show that the improvement in squared-error risk (1) when one of the regions R_m is split into two daughter regions, where n is the number of observations in the parent R_m, n_l, n_r the numbers respectively in the left and right daughters, and $\bar{y_l}$ and $\bar{y_r}$ are the means of the outcome variable y for observations in the respective daughter regions.

To show this, we first define some notation. Let:

$$\begin{aligned} \bar{m} &= \{k \mid x_k \in R_m\} \\ \bar{l} &= \{k \mid x_k \in R_l\} \\ \bar{r} &= \{k \mid x_k \in R_r\} \end{aligned}$$

and note that this implies:

$$\bar{l} \cup \bar{r} = \bar{m}$$

Now, we can define the squared-error risk from the full region R_m and the squared-error risk from the daughter regions R_l and R_r as:

$$\begin{aligned} S_m &= \sum_{i \in \bar{m}} (y_i - \bar{y}_m)^2 \\ S_{lr} &= \sum_{i \in \bar{l}} (y_i - \bar{y}_l)^2 + \sum_{i \in \bar{r}} (y_i - \bar{y}_r)^2 \end{aligned}$$

where

$$\begin{aligned} \bar{y}_m &= \frac{1}{n} \sum_{i \in \bar{m}} y_i \\ \bar{y}_l &= \frac{1}{n_l} \sum_{i \in \bar{l}} y_i \\ \bar{y}_r &= \frac{1}{n_r} \sum_{i \in \bar{r}} y_i \end{aligned}$$

Note that from the above equations, we can derive the following result:

$$\begin{aligned} & \bar{y}_m = \frac{1}{n} \sum_{i \in \bar{m}} y_i \\ \implies & \bar{y}_m = \frac{1}{n} (\sum_{i \in \bar{l}} y_i + \sum_{i \in \bar{r}} y_i) \\ \implies & \bar{y}_m = \frac{n_l}{n} \bar{y}_l + \frac{n_r}{n} \bar{y}_r \end{aligned}$$

We are interested in the difference between S_m and S_{lr}. Note that this will give the total improvement in the squared-error risk because the risk associated with other regions $\neq m$ is not affected by the splitting of region m into two daughter regions.

We begin by rewriting the squared-error risks as follows:

$$\begin{aligned} S_m &= \sum_{i \in \bar{m}} (y_i - \bar{y}_m)^2 \\ &= \sum_{i \in \bar{m}} y_i^2 - 2\bar{y}_m \sum_{i \in \bar{m}} y_i \ + \sum_{i \in \bar{m}} \bar{y}_m^2 \\ &= \sum_{i \in \bar{m}} y_i^2 \ - 2n\bar{y}_m^2 \ + n\bar{y}_m^2 \\ &= \sum_{i \in \bar{m}} y_i^2 \ - n\bar{y}_m^2 \end{aligned}$$

Similarly:

$$\begin{aligned} S_{lr} &= \sum_{i\in\bar{l}}(y_i-\bar{y}_l)^2 \ + \sum_{i\in\bar{r}}(y_i-\bar{y}_r)^2 \\ &= \sum_{i\in\bar{l}} y_i^2 \ - n_l\bar{y}_l^2 \ + \sum_{i\in\bar{r}} y_i^2 \ - n_r\bar{y}_r^2 \\ &= \sum_{i\in\bar{m}} y_i^2 \ - n_l\bar{y}_l^2 \ - n_r\bar{y}_r^2 \end{aligned}$$

Then, the improvement in the squared-error risk from splitting R_m into the two daughter regions R_l and R_r is given by:

$$\begin{aligned} S_m \ - S_{lr} &= (\sum_{i\in\bar{m}} y_i^2 \ - n\bar{y}_m^2) \ - (\sum_{i\in\bar{m}} y_i^2 \ - n_l\bar{y}_l^2 \ - n_r\bar{y}_r^2) \\ &= -n\bar{y}_m^2 \ + n_l\bar{y}_l^2 \ + n_r\bar{y}_r^2 \\ &= -n(\frac{n_l}{n}\bar{y}_l \ + \frac{n_r}{n}\bar{y}_r)^2 \ + n_l\bar{y}_l^2 \ + n_r\bar{y}_r^2 \\ &= -n(\frac{n_l^2}{n^2}\bar{y}_l^2 \ + \ \frac{2n_ln_r}{n^2}\bar{y}_l\bar{y}_r \ + \ \frac{n_r^2}{n^2}\bar{y}_r^2) \ + n_l\bar{y}_l^2 \ + n_r\bar{y}_r^2 \\ &= -\frac{n_l^2}{n}\bar{y}_l^2 \ - \ \frac{2n_ln_r}{n}\bar{y}_l\bar{y}_r \ - \ \frac{n_r^2}{n}\bar{y}_r^2 \ + n_l\bar{y}_l^2 \ + n_r\bar{y}_r^2 \\ &= (\frac{nn_l-n_l^2}{n}\bar{y}_l^2) \ + (\frac{nn_r-n_r^2}{n}\bar{y}_r^2) - \ \frac{2n_ln_r}{n}\bar{y}_l\bar{y}_r \\ &= [\frac{(n_l+n_r)n_l-n_l^2}{n}\bar{y}_l^2] \ + [\frac{(n_l+n_r)n_r-n_r^2}{n}\bar{y}_r^2] - \ \frac{2n_ln_r}{n}\bar{y}_l\bar{y}_r \\ &= \frac{n_ln_r}{n}\bar{y}_l^2 \ + \frac{n_ln_r}{n}\bar{y}_r^2 - \ \frac{2n_ln_r}{n}\bar{y}_l\bar{y}_r \\ &= \frac{n_ln_r}{n}(\bar{y}_l^2-2\bar{y}_l\bar{y}_r+\bar{y}_r^2) \\ &= \frac{n_ln_r}{n}(\bar{y}_l-\bar{y}_r)^2 \qquad \blacksquare \end{aligned}$$

#Question 13 ###Derive an updating formula for calculating the change in the improvement in prediction risk as the result of a split when the split is modified by one observation changing sides. Suppose that we have an existing split of R_m into R_l and R_r, with n, n_l, n_r as well as $\bar{y}_l$ and $\bar{y}_r$ defined as in the question 12. \ Let y^* be the outcome of the one observation that is changing sides. Without loss of generality, we can assume that the observation belonged to R_l, and after the new switch will belong to R_n. Then the improvement from this swap can be written as:

$$\begin{aligned} Improvement &= S_m - S_{lr} - (S_m - S_{lr\_new}) \\ &= \frac{n_ln_r}{n}(\bar{y}_l-\bar{y}_r)^2 - \frac{(n_l-1)*(n_r+1)}{n}\big(\frac{n_l(\bar{y}_l-y^*)}{n_l-1} - \frac{n_r\bar{y}_r+y^*}{n_r+1}\big)^2 \end{aligned}$$

#Question 14 ###Is enlarging your function class F always a good idea? Will it necessarily lead to better expected mse on future data? Why or why not? Conversely, is it always better to reduce the size of F (increasing the restriction on $g(x)$), thereby fitting the training data less well? Why or why not?

No it is not always a good idea to increase the size of the function class. This can often lead to over-fitting and an increase in variance. One example is found in the class of polynomial functions. We can reduce MSE by allowing higher and higher order polynomials in F, but the result on new data may be worse than even a simple first order polynomial fit (linear model). The higher order polynomial will have many curves and pass through all the data points, but it will not generalize well on future data. In other words, because $mse = bias^2 + variance$, if the decrease in bias from the higher order models does not compensate for the increase in variance, the mse on future data might be worse. Therefore, it is not always a good idea to allow larger and larger function classes for F.

Conversely, it is not always a better idea to reduce the size of F, increasing the bias of the model. If the function class F is too restrictive, then the estimate of the target function that is found may be nowhere close to the true target function. In this case, though the model may have low variance, it will not necessarily lead to a low mse on future data.

#Question 15 ###The recursive partitioning strategy described in class for building decision trees uses two-way (binary) splits at each step. This is not fundamental, and one could envision multi-way splits of each non-terminal node creating several (rather than two) daughter regions with each split. What would be the relative advantages and disadvantages of a such a multi-way splitting strategy?

This is not a good general strategy. One problem is that multi-way splits fragment the data too quickly, leaving insufficient data at the next level down. Hence we would want to use such splits only when needed. Another problem is that multi-way splits make the assumption that the "next-best" binary split occurs in the same variable x_j as the first split. This is an unnecessary restriction. Since multi-way splits can be achieved by a series of binary splits anyway, the latter are preferred.

The possible advantages of a multi-way splitting approach are:

- Increased interpretability. By splitting multiple ways on a single variable, we avoid convoluted and unnecessarily deep tree structures where the pathway from a node to leaf contains multiple splits on the same attribute.
- Ability to apply domain knowledge. If for some reason we had prior knowledge that a multiway split would lead to the best outcome for a particular variable, enforcing this could lead to a better tree than the greedy recursive partitioning approach.

#Question 16 ###What would be the advantages and disadvantages of including linear combination splits in the tree building strategy?

The primary disadvantages of including linear combination splits in the tree building strategy would be that the resulting tree is less interpretable by a user. In addition, building the tree would require a much more difficult search strategy and be more computationally intensive. Finally, there would be more variance in the solution due to the larger function class.

The advantage of including linear combination splits in tree building would be that it would better capture interactions between features.