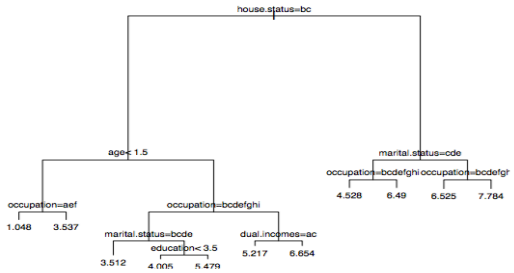
Problem 1

First build a large tree so we can use rpart's cross-validation feature to determine the optimal tree size. The cross-validation errors can be found using the function **printcp** and used to find the optimal value of the complexity parameter, which for us is about .0007. Some of the R code required is given below.

```
income.data = read.csv("Income_Data.txt", header = FALSE)
names = c("income", "sex", "marital.status", "age",
    "education", "occupation", "time.in.area", "dual.incomes",
    "n.in.household","n.children", "house.status", "house.type",
    "ethnicity", "language")
colnames(income.data) = names
factors = c(2,3,6,8,11,12,13,14)
income.data[,factors] = lapply(income.data[,factors], factor)
tree = rpart(income ~ ., data = income.data,
    control = rpart.control(cp = .0001))
tree.opt = rpart(income ~ ., data = income.data,
    control = rpart.control(cp = .0007))
```

The tree is a little bit too big to plot, so here is a plot of a smaller tree.

```
par(mar = rep(1,4))
tree.pruned.more = prune(tree.pruned, cp = .006)
plot(tree.pruned.more); text(tree.pruned.more, cex = .7)
```



The first split is on house status — if you own your house, your income is likely to be higher. Subsequent splits are on age, marital status, and occupation. Young people are likely to have low incomes, people who are married or living together are likely to have a higher household income than those who are divorced or separated, widowed, or single (makes sense because people living together will have two incomes instead of one), and people whose occupation is professional/managerial are likely to have a higher income than others.

(a) Surrogate splits were used in the construction of the optimal tree that I obtained. A surrogate split is used when the variable normally used to determine which branch of the tree to traverse is missing. Surrogate splitting calculates a set of surrogate variables that can be used to estimate which branch of the tree we should traverse when the primary variable is missing. There are several surrogate splits in the tree found above. For example, in the first node of the tree, the split is on householder status, but there are surrogate splits on age, marital status, dual incomes, occupation, and education. See the output below.

```
summary(prune(tree, cp = .2))
```

```
## Call:
## rpart(formula = income ~ ., data = income.data, control = rpart.control(cp = 1e-04
##   n= 8993
##
##         CP nsplit rel error    xerror        xstd
```

```
## 1 0.223922      0  1.000000 1.0001783 0.008259014
## 2 0.200000      1  0.776078 0.7763418 0.008852116
##
## Variable importance
##   house.status            age marital.status   dual.incomes     occupation
##             45             17             15             14              6
##      education
##              2
##
## Node number 1: 8993 observations,    complexity param=0.223922
##   mean=4.895029, MSE=7.692528
##   left son=2 (5646 obs) right son=3 (3347 obs)
##   Primary splits:
##       house.status   splits as  RLL,       improve=0.2243474, (240 missing)
##       age            < 2.5 to the left,    improve=0.2121074, (0 missing)
##       marital.status splits as  RLLLL,     improve=0.2060636, (160 missing)
##       dual.incomes   splits as  LRR,       improve=0.1915465, (0 missing)
##       occupation     splits as  RLLLRLLRL, improve=0.1732841, (136 missing)
##   Surrogate splits:
##       age            < 3.5 to the left,    agree=0.769, adj=0.378, (240 split)
##       marital.status splits as  RLLRL,     agree=0.755, adj=0.341, (0 split)
##       dual.incomes   splits as  LRR,       agree=0.742, adj=0.307, (0 split)
##       occupation     splits as  RLLLRLLRL, agree=0.681, adj=0.143, (0 split)
##       education      < 5.5 to the left,    agree=0.647, adj=0.051, (0 split)
##
## Node number 2: 5646 observations
##   mean=3.88452, MSE=6.656165
##
## Node number 3: 3347 observations
##   mean=6.599641, MSE=4.812525
```

(b) We can used the pruned tree to predict my income.

```
new.obs = c(NA, 2, 5, 3, 6, 6, 3, 1, 1, 0, 2, 3, 8, 1)
income.data[nrow(income.data) + 1, ] = new.obs
income.data[nrow(income.data),]
```

```
##      income sex marital.status age education occupation time.in.area
## 8994     NA   2              5   3         6          6            3
##      dual.incomes n.in.household n.children house.status house.type
## 8994            1               1          0            2          3
##      ethnicity language
## 8994         8        1
```

```
predict(tree.pruned, income.data[nrow(income.data), ])
```

```
##     8994
## 2.564184
```

Bin 2 for income is \$10,000-\$15,000, and bin 3 for income is \$15,000 to \$20,000 so this corresponds to something around \$15,000.

Problem 2

The aim of this problem is to fit a classification tree to the housing data using the R package *rpart*. We thus begin by loading the data, and indicating which variables are categorical (note that some of them are ordinal variables, and we should treat them as such!)

```
library(rpart)
info=c("Y","sex","marital_status","age","education","occupation","annual_income","length",
  "dual_incomes","size_household","nb_minors","householder_status","ethnic","language")
X=read.table("Housetype_Data.txt",sep=",",header=F)
Y=X[,1]
X=data.frame(X)
colnames(X)=info
X$Y<-as.factor(X$Y)
X$sex<-as.factor(X$sex)
X$marital_status<-as.factor(X$marital_status)
X$occupation<-as.factor(X$occupation)
X$dual_incomes<-as.factor(X$dual_incomes)
X$ethnic<-as.factor(X$ethnic)
X$householder_status<-as.factor(X$householder_status)
X$language<-as.factor(X$language)
#### The goal is too infer Y from X by fitting an  optimal classification tree
```

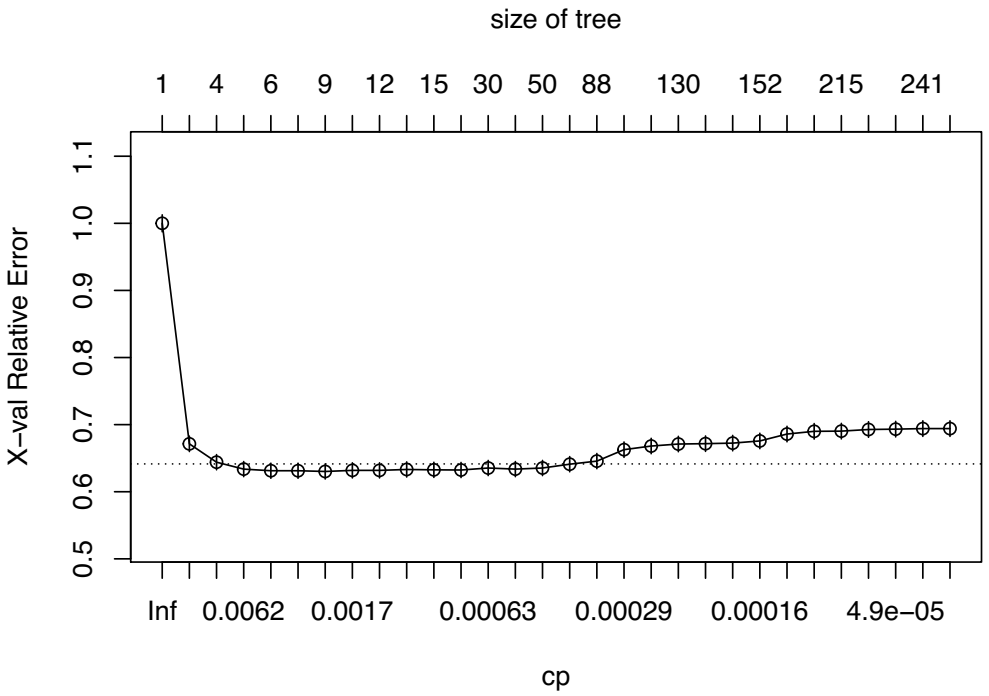
We can begin by analyzing the data: nb of missing values, spread, etc (just to get the feel of the data)

```
##### We begin by analyzing the data: nb of missing values, spread, etc
K= 5 ## number of possible classes for the housing
summary(X)
```

```
##  Y        sex      marital_status      age          education
##  1:5319   1:4043   1   :3351      Min.   :1.000   Min.   :1.000
##  2: 674   2:4970   2   : 667      1st Qu.:2.000   1st Qu.:3.000
##  3:2454            3   : 856      Median :3.000   Median :4.000
##  4: 162            4   : 312      Mean   :3.423   Mean   :3.831
##  5: 404            5   :3669      3rd Qu.:4.000   3rd Qu.:5.000
##                    NA's: 158      Max.   :7.000   Max.   :6.000
##                                                   NA's   :93
##    occupation   annual_income       length    dual_incomes size_household
##  1      :2798   Min.   :1.000   Min.   :1.0   1:5439       Min.   :1.000
##  6      :1475   1st Qu.:2.000   1st Qu.:4.0   2:2217       1st Qu.:2.000
##  4      :1049   Median :5.000   Median :5.0   3:1357       Median :3.000
##  2      : 786   Mean   :4.929   Mean   :4.2                Mean   :2.874
##  3      : 755   3rd Qu.:7.000   3rd Qu.:5.0                3rd Qu.:4.000
##  (Other):1994   Max.   :9.000   Max.   :5.0                Max.   :9.000
##  NA's   : 156   NA's   :377     NA's   :942                NA's   :331
##    nb_minors      householder_status     ethnic     language
##  Min.   :0.0000   1   :3304          7      :5845   1   :7803
##  1st Qu.:0.0000   2   :3664          5      :1235   2   : 591
##  Median :0.0000   3   :1851          3      : 893   3   : 260
##  Mean   :0.6767   NA's: 194          2      : 471   NA's: 359
##  3rd Qu.:1.0000                      8      : 238
##  Max.   :9.0000                      (Other): 270
##                                      NA's   :  61
```

We now try to grow the tree: we begin to set the Complexity paramter as low as possible to grow as "maximum tree". We will then prune the tree choosing the adequate Cp value using the 1MSE rule:we choose the most parsimonious model whose error is no more than one standard error above the error of the best mode.

```
# grow tree
fit <- rpart(Y~ .,method="class", data=X,cp=0) ### Y vs X (the rest of the variables)
### This  method does exactly what we want: it gives back a classification tree according to the formula
### The complexity paramter
plotcp(fit) # visualize cross-validation results
```




```
### Prune the tree: select the CP that is 1MSE above the cp-value that minimizes the misclassification e
# get index of CP with lowest xerror
opt_cp <- which.min(fit$cptable[,'xerror'])
candidate_cp<- which(fit$cptable[,'xerror']>(fit$cptable[opt_cp,'xerror']+fit$cptable[opt_cp,'xstd']))
cp_mse_ind<- min(which(fit$cptable[,'xerror']<(fit$cptable[opt_cp,'xerror']+fit$cptable[opt_cp,'xstd'])))
cp_mse=fit$cptable[cp_mse_ind,'CP']
#get its value
#prune tree
pruned_model <- prune(fit,cp_mse)
### as input
printcp(pruned_model) # display the results in terms of complexity paramter
```

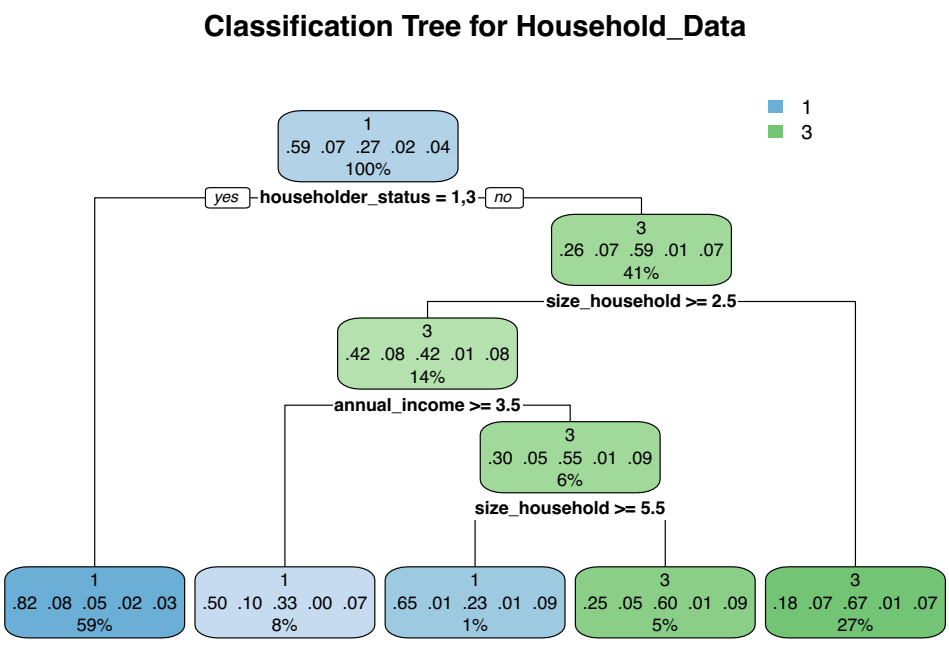
```
##
## Classification tree:
## rpart(formula = Y ~ ., data = X, method = "class", cp = 0)
##
## Variables actually used in tree construction:
```

```
## [1] annual_income      householder_status size_household
##
## Root node error: 3694/9013 = 0.40985
##
## n= 9013
##
##          CP nsplit rel error  xerror     xstd
## 1 0.3286410      0   1.00000 1.00000 0.012640
## 2 0.0167840      1   0.67136 0.67136 0.011478
## 3 0.0078506      3   0.63779 0.64402 0.011328
## 4 0.0048728      4   0.62994 0.63373 0.011269
```

```
#plot tree
library(rpart.plot)
```

```
## Warning: package 'rpart.plot' was built under R version 3.3.2
```

```
rpart.plot(pruned_model, uniform=TRUE,main="Classification Tree for Household_Data")
```



We note that:

- class 2 (Condominium) and 4 (mobile home) are not predicted by our classification tree. This is due to the fact that these are relatively rare classes, which account for only respectively 7.5 and 1.8 % of the observations.
- The most important variable in our case if the householder status (own, rent, live with parent)
- The misclassification cross-validated error rate is roughly $0.409 \times 0.636 = 0.2601$

Problem 3

- Overfitting. If we have a very flexible model and not enough data, we can fit the training data well, but not able to generalize well to future data.
- Training data might not be a good representation of the population.

Problem 4

(Answer partially inspired by Stephen Bates, Varun Gupta)

A statistical issue is **overfitting**: the function that best fits our data within the space of all functions will have very high variance. Also, if all the X values are distinct, there is an infinite number of functions that perfectly interpolate the data. Therefore, the solution is **not unique**. Furthermore, searching the space of all functions is **not algorithmically tractable**.

Problem 5

(Answer partially inspired by Stephen Bates, Varun Gupta)

The target function is the function that minimizes the population risk:

$$F^* = \text{argmin}_F \, \mathbb{E}_{XY}[L(Y, F(X))]$$

The target function is the best function of the available predictors, according to this criterion. It is possible that our predictors do not contain all the information required to perfectly predict the response. As an extreme exaple, it may happen that the response is independent (or nearly independent) of all available predictors.

Problem 6

The population risk associated with F is

$$R(F) = \mathbb{E}_{XY}[L(Y, F(X))]$$

Since the joint distribution of (X, Y) is unknown, we evaluate it instead on the training data and thus obtain an empirical risk,

$$\hat{R}(F) = \frac{1}{n} \sum_{i=1}^{n} L(Y_i, F(X_i))$$

The empirical risk will usually be an optimistic estimate for the actual risk, because during the learning process, the empirical loss is both used to evaluate the prediction error and adjust the prediction F to minimize it.

However, the higher the signal to noise ratio is and the higher the number of training samples, the better surrogate it will be for the population risk.

Problem 7

The formula for the misclassification risk is

$$\text{Risk} = \sum_{k=1}^{K} \sum_{\ell=1}^{K} \Pr[y = c_\ell, \hat{y} = c_k] L_{kl}.$$

According to the problem, we can assume that $L_{kl} = r$ for some constant $r > 0$ for all $k \neq \ell$. Furthermore, we know that $L_{kk} = 0$ because the loss is zero for incorrect classification. Using these two facts, we get

$$\begin{aligned} \text{Risk} &= \sum_{k=1}^{K} \sum_{\ell=1}^{K} \Pr[y = c_\ell, \hat{y} = c_k] r 1(k \neq l) \\ &= r \sum_{k=1}^{K} \sum_{\ell=1}^{K} \Pr[y = c_\ell, \hat{y} = c_k] 1(k \neq l) \\ &= r \Pr[y \neq \hat{y}]. \end{aligned}$$

which is a constant times the error rate.

In the general case, the Bayes prediction rule for a new observation x is to predict $\hat{y} = c_k$, where

$$k = \text{argmin}_k \sum_{\ell=1}^{K} \Pr(y = c_\ell | x) L_{kl}.$$

However, if $L_{kl} = 1(k \neq \ell)$, then the above simplifies to

$$k = \text{argmax}_k \Pr(y = c_k | x),$$

or equivalently,

$$k = \text{argmax}_k p(x | y = c_k) \pi(y).$$

In other words, the Bayes prediction rule chooses the class with the highest posterior probability in the special case of zero-one loss.

Problem 8

(Thanks to Stephen Bates) No. The Bayes rule will choose to classify observations into whichever class has the highest probability, so the accuracy only depends on how well we are identifying classes of high probability. There are many probabilities of the classes that correspond to the same Bayes rule, but the probabilities can be very different.

Problem 9

In general, the bias-variance trade-off refers to the phenomenon that, in statistical prediction problems, (the absolute value of) bias decreases/increases and variance increases/decreases as the model complexity increases/decreases, respectively. Suppose we have a small function class, and we expand it to make it bigger and bigger. Then, the distance from the prediction to the target function decreases, which means that the bias decreases. On the other hand, it becomes increasingly harder to find the function in the class closest to the target function because there are larger number of functions that we can choose from. This means that the variance increases. Since bias and variance are two sources of error, it is important to choose the parameter carefully (for example, by cross validation) to balance the trade-off between them.

Problem 10

(Answer partially inspired by Evan Rosenman, Rina Friedberg, Alex Chin, Stephen Bates, Varun Gupta)

We are being asked why we do not order the remaining variables based on how well they predict the primary split variable, as opposed to the response, when the former is missing. Note that this is the only interpretation of this question that does not lead to a completely trivial answer.

The alternative approach that is being suggested would be equivalent to splitting on the missing values as a separate category and then using the surrogate to further split on the "missing" side of the previous split, and the primary variable on the "not missing" side. This creates a more complex procedure that will have higher variance than using the surrogate splits to predict the missing primary variables.

Problem 11

For a given data set $\{(x_i, y_i)\}_{i=1,\ldots,N}$, the squared-error risk is given by

$$g(c_1, \ldots, c_m) := \sum_{i=1}^{N} [y_i - F(x_i)]^2 = \sum_{m=1}^{M} \sum_{i=1}^{N} I(x_i \in R_m)(y_i - c_m)^2,$$

because $\{R_m\}_{m=1,\cdots,M}$ are disjoint. Its partial derivative with respect to c_m, $m = 1, \ldots, M$, is

$$\frac{\partial g}{\partial c_m} = 2 \sum_{i=1}^{N} I(x_i \in R_m)(c_m - y_i) = 2 \left(\sum_{i=1}^{N} I(x_i \in R_m) c_m - \sum_{i=1}^{N} I(x_i \in R_m) y_i \right).$$

Assuming there exists at least one data point whose predictor part is contained in R_m, i.e. $\sum_{i=1}^{N} I(x_i \in R_m) > 0$ (which is the case we are interested in), from the definition of $\hat{c}_m$ given in the problem we get

$$\frac{\partial g}{\partial c_m} = 2 \sum_{i=1}^{N} I(x_i \in R_m)(c_m - \hat{c}_m).$$

Hence, g is strictly increasing (decreasing) when $c_m > \hat{c}_m$ ($c_m < \hat{c}_m$), where $\hat{c}_m$ is as given in the problem. This shows that $\hat{c}_m$ is the minimizer of the squared-error risk. □

Problem 12

There is no change in any region except for R_m, which is split into R_m^l and R_m^r. That is, the regression surface is unchanged outside of R_m, and within this region, the surface has been updated to be the average within each of R_m^l and R_m^r, rather than the average of R_m overall. This means that the change in the left region is

$$\sum_{i \in R_m^l} \left[(y_i - \bar{y})^2 - \left(y_i - \bar{y}^l\right)^2 \right],$$

where $\bar{y}$ denotes the average over all y's in R_m and $\bar{y}^l$ and $\bar{y}^r$ are averages over R_m^l and R_m^r, respectively. The first squared term is the approximation error before the split, while the second is the error afterwards. An analogous expression holds for R_m^r.

To begin simplifying this expression, expand the squares and use the fact that $\sum_{i=1}^n x_i = n\bar{x}$,

$$\begin{aligned}
\sum_{i \in R_m^l} \left[(y_i - \bar{y})^2 - \left(y_i - \bar{y}^l\right)^2 \right] &= \sum_{i \in R_m^l} \left[y_i^2 - 2y_i\bar{y} + \bar{y}^2 - y_i^2 + 2y_i\bar{y}^l - (\bar{y}^l)^2 \right] \\
&= -2n_l\bar{y}^l\bar{y} + n_l\bar{y}^2 + n_l(\bar{y}^l)^2 \\
&= n_l\left(\bar{y}^l - \bar{y}\right)^2
\end{aligned}$$

Therefore, the overall change across both R_m^l and R_m^r is

$$n_l\left(\bar{y}^l - \bar{y}\right)^2 + n_r\left(\bar{y}^r - \bar{y}\right)^2. \tag{1}$$

To get rid of the $\bar{y}$ appearing in this expression, note that

$$\bar{y} = \frac{1}{n}\left(n_r\bar{y}^r + n_l\bar{y}^l\right),$$

and hence

$$\begin{aligned}
\bar{y}^l - \bar{y} &= \bar{y}^l - \frac{1}{n}\left(n_r\bar{y}^r + n_l\bar{y}^l\right) \\
&= \frac{n_r}{n}\left(\bar{y}^l - \bar{y}^r\right),
\end{aligned}$$

and

$$\bar{y}^r - \bar{y} = \frac{n_l}{n}\left(\bar{y}^r - \bar{y}^l\right).$$

Substituting into equation 1 yields

$$\begin{aligned}
n_l\left(\frac{n_r}{n}\right)^2\left(\bar{y}^l - \bar{y}^r\right)^2 + n_r\left(\frac{n_l}{n}\right)^2\left(\bar{y}^r - \bar{y}^l\right)^2 &= \frac{n_l n_r^2 + n_r n_l^2}{n^2}\left(\bar{y}^r - \bar{y}^l\right)^2 \\
&= \frac{n_l n_r\left(n_r + n_l\right)}{n^2}\left(\bar{y}^r - \bar{y}^l\right)^2 \\
&= \frac{n_l n_r}{n}\left(\bar{y}^r - \bar{y}^l\right)^2,
\end{aligned}$$

as needed.

Problem 13

(Thanks to Feng Ruan, Jiafan Yu, Junyang Qian, Yu Bai)

Suppose the part of response data that arrives at a particular node is $\{y_i\}_{i=1}^n$. The part that goes to the left child is $\{y_i\}_{i\in\mathcal{L}}$ and right child is $\{y_i\}_{i\in\mathcal{R}}$. Denote the corresponding means as $\bar{y}, \bar{y}_L$ and $\bar{y}_R$, the original improvement as $\hat{I}_m$ and that after modification as $\hat{I}'_m$. Then,

$$\begin{aligned} N\hat{I}_m &= \sum_{i=1}^n (y_i-\bar{y})^2 - \sum_{i\in\mathcal{L}}(y_i-\bar{y}_L)^2 - \sum_{i\in\mathcal{R}}(y_i-\bar{y}_R)^2 \\ &= \left(\sum_{i=1}^n y_i^2 - n(\bar{y})^2\right) - \left(\sum_{i\in\mathcal{L}} y_i^2 - n_L(\bar{y}_L)^2\right) - \left(\sum_{i\in\mathcal{R}} y_i^2 - n_R(\bar{y}_R)^2\right) \\ &= n_L(\bar{y}_L)^2 + n_R(\bar{y}_R)^2 - n(\bar{y})^2. \end{aligned}$$

Without loss of generality, we assume one point y_0 is moved from the left child to the right child. After modification, $\bar{y}' = \bar{y}$,

$$\bar{y}'_L = \frac{n_L\bar{y}_L - y_0}{n_L-1}, \quad \bar{y}'_R = \frac{n_R\bar{y}_R + y_0}{n_R+1}.$$

Then we have

$$\begin{aligned} N\hat{I}'_m &= (n_L-1)(\bar{y}'_L)^2 + (n_R+1)(\bar{y}'_R)^2 - n(\bar{y})^2 \\ &= \frac{(n_L\bar{y}_L - y_0)^2}{n_L-1} + \frac{(n_R\bar{y}_R+y_0)^2}{n_R+1} - n(\bar{y})^2. \end{aligned}$$

Thus the change in improvement

$$\begin{aligned} N\hat{I}'_m - N\hat{I}_m &= \frac{(n_L\bar{y}_L - y_0)^2}{n_L-1} + \frac{(n_R\bar{y}_R+y_0)^2}{n_R+1} - n_L(\bar{y}_L)^2 - n_R(\bar{y}_R)^2 \\ &= \frac{(n_L\bar{y}_L - y_0)^2 - (n_L-1)n_L(\bar{y}_L)^2}{n_L-1} \\ &\quad + \frac{(n_R\bar{y}_R+y_0)^2 - (n_R+1)n_R(\bar{y}_R)^2}{n_R+1} \\ &= \frac{n_L(\bar{y}_L)^2 - y_0(2n_L\bar{y}_L - y_0)}{n_L-1} + \frac{-n_R(\bar{y}_R)^2 + y_0(2n_R\bar{y}_R+y_0)}{n_R+1} \\ &= y_0^2\left(\frac{1}{n_L-1}+\frac{1}{n_R+1}\right) + y_0\left(-\frac{2n_L\bar{y}_L}{n_L-1} + \frac{2n_R\bar{y}_R}{n_R+1}\right) \\ &\quad + \left(\frac{n_L}{n_L-1}(\bar{y}_L)^2 - \frac{n_R}{n_R+1}(\bar{y}_R)^2\right). \end{aligned}$$

Thus the change of improvement in the risk is

$$\hat{I}'_m - \hat{I}_m = \frac{1}{N}\left(y_0^2\left(\frac{1}{n_L-1}+\frac{1}{n_R+1}\right) + y_0\left(-\frac{2n_L\bar{y}_L}{n_L-1}+\frac{2n_R\bar{y}_R}{n_R+1}\right) + \left(\frac{n_L}{n_L-1}(\bar{y}_L)^2 - \frac{n_R}{n_R+1}(\bar{y}_R)^2\right)\right).$$

Problem 14

The choice of whether to enlarge or restrict the class of F with which to define the target function is related to the bias-variance tradeoff. A larger class F allows for fits with lower bias, but in the absence of a sufficient amount of training data, the estimates $\hat{f}$ may be highly variable. Further, in the case that the underlying response function belongs to a restricted class, estimation using that restricted class will be more statistically efficient.

It is important to evaluate the quality of the fit $\hat{f}$ using test data – improvement in mse on the training data alone is not enough. Whether or not increasing or reducing the size of F is a good idea depends on the presence of enough data to fit and evaluate $\hat{f}$ reliably (more data justifies the use of a richer class F) as well as any apriori knowledge about the complexity of the underlying response function (if it is simple, it is better to use a smaller class F containing plausible response functions).

Problem 15

The advantage of multi-way splitting is that it can approximate more complicated function not well approximted by two way splitting. However, the distanvantage is that it can split the data too quickly and leaving insufficients data at the next level down.

Problem 16

The advantages of such a model include:

- Invariance to linear transformations of the input
- May in some situations be a better fit to the true function

However, some disadvantages are:

- Increases the order of the computation in finding the best split from order $O(p)$ to order exponential in p.
- Less fine-grained control over bias-variance tradeoff, because the complexity increases greatly at each split.
- More difficult to come up with a strategy for dealing with missing variables, since surrogate splitting strategy cannot be applied.
- More difficult to interpret the resulting trees.

Fun fact: (not expected as part of the answer.) Such trees are known as "oblique decision trees" or decision trees with linear decision rules. They were included in the book that introduced CART (Breiman, L., Friedman, J., Stone, C. J., Olshen, R. A., 1984. Classification and regression trees. CRC press.), and

have been revisited periodically in the machine learning literature since then. But they do not appear to be widely used in applications.