MATH 342W / 642 / RM 742 Spring 2025 HW #5

Sergio E. Garcia Tapia

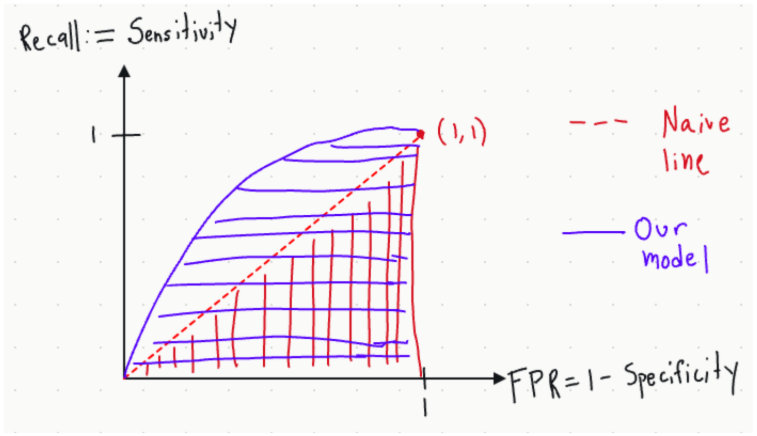
Friday 16th May, 2025

Problem 1

These are some questions related to probability estimation modeling and asymmetric cost modeling. It might be helpful to look back to the review the logistic regression before doing this problem.

(a) [easy] Graph a canonical ROC and label the axes. In your drawing estimate AUC. Explain very clearly what is measured by the x axis and the y axis.

See the following figure:



In the plot, the x-axis is the *False Positive Rate* (FPR), which is given by

$$FPR := \frac{FP}{N}$$

where FP is the number of false positive predictions by a classification model and N is the total number of "negative" (0) responses in the phenomenon under consideration. The FPR answers the question, "of the values our model predicts as negative, what proportion are incorrect?" The y-axis corresponds to *Recall*, given by

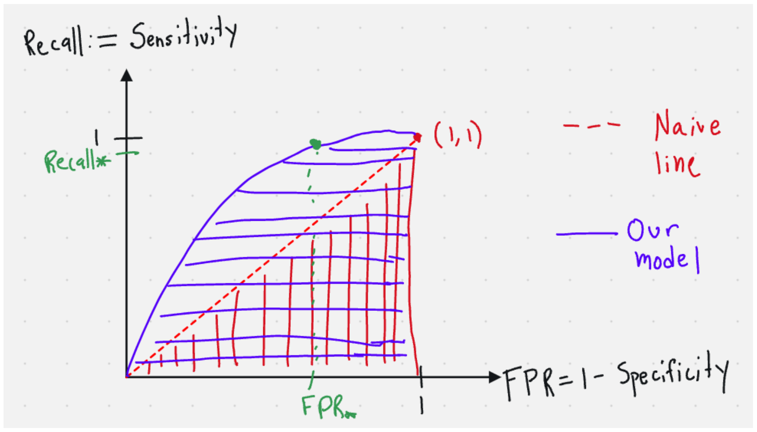
$$\text{Recall} := \frac{TP}{P}$$

Recall answers the question "what proportion of the positive responses did we successfully predict correctly?". Each point on the ROC curve is a (FPR, Recall) pair, and each value of that pair is a function of p_{th}, the probability threshold used for prediction. Therefore, the curve actually shows several binary classification models for a given probability estimation model (one for each value fo p_{th}).

In the plot, the dotted (red) line represents a naive model, which corresponds to probability predictions made by picking values from a uniform distribution on $(0, 1)$. The area under the curve (AUC) for the naive model is 0.5. In purple with a solid line is "our model", which we are assuming outperforms the naive model. The area under the curve is probably around 0.75.

(b) [easy] Pick one point on your ROC curve from the previous question. Explain a situation why you would employ this model.

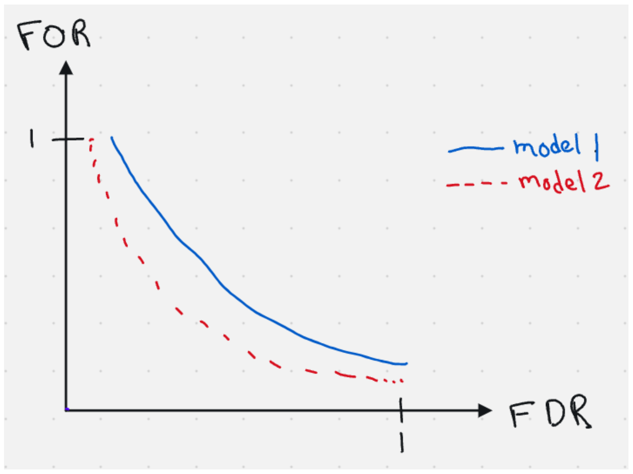
See the following figure:



I picked a point (FPR_*, Recall_*), which corresponds to a model built with some $p_{\text{th}*}$ value that exhibits high Recall, and relatively high FPR. A high Recall model implicitly weighs false negatives as more costly. Imagine designing a fire alarm system that attempts to detect whether there is a fire in a building. A false positive means the alarm goes off and the firefighters are called in, but there is no actual fire. A false negative means the alarm does not detect a fire, but there is in fact a fire. In this case false negatives are more costly because human lives are at stake.

(c) [harder] Graph a FDR-FOR curve and label the axes. Explain very clearly what is measured by the x axis and the y axis.

See the following figure:



In the plot, the x-axis is the *False Discovery Rate*, given by

$$FDR := \frac{FP}{PP}$$

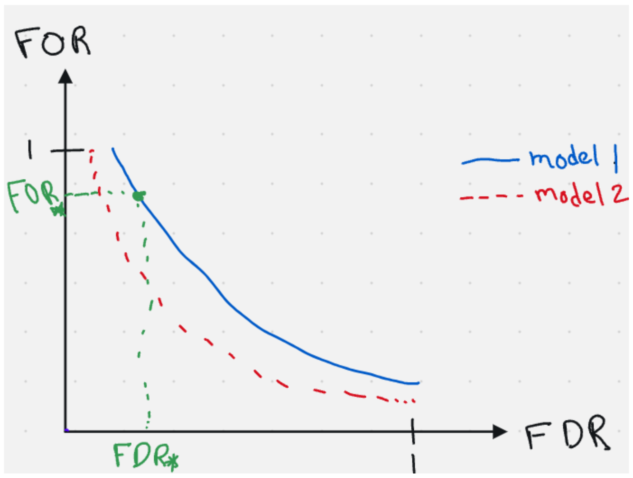
where FP is the number of false positives and PP is the number of predicted positives. FDR measures the rate of mistakes in the positive predictions. In the y-axis is the *False Omission Rate*, given by

$$FOR := \frac{FN}{PN}$$

where FN is the number of false negatives and PN is the number of predicted negatives. FOR measures the rate of mistakes in the negative predictions. Each point on the FDR-FOR curve corresponds to a different p_{th} value, th e probability threshold used for prediction. The plot shows two curves. and each curve corresponds to a single probability estimation model, but several binary classification models (one for each value of p_{th}). The curves show that there is a trade-off between FDR and FOR. As p_{th} increases, FDR decreases while FOR increases.

(d) [easy] Pick one point on your FDR-FOR curve from the previous question. Explain a situation why you would employ this model.

See the following figure:



I picked a point corresponding to a high FOR and a low FDR. This might make sense in a situation where the cost of false positives is prohibitive in comparison with the cost of false negatives. Suppose, for example, a secured facility that uses biometric information to determine whether to allow a person to enter. A false positive in this situation means letting uncleared personnel enter the facility, while a false negative may be an inconvenience to someone who actually works there but can be resolved. The cost of false positives is high because it puts the security of the facility at risk.

(e) [difficult] [MA] The line of random guessing on the ROC curve is the diagonal line with slope one extending from the origin. What is the corresponding line of random guessing in the FDR-FOR curve? This is not easy...

Problem 2

These are some questions related to bias-variance decomposition. Assume the two assumptions from the notes about the random variable model that produces the δ values, the error due to ignorance.

(a) [easy] Write down (do not derive) the decomposition of MSE for a given $\boldsymbol{x}_*$ where $\mathbb{D}$ is assumed fixed but the response associated with $\boldsymbol{x}_*$ is assumed random.

The MSE for a given $\boldsymbol{x}_*$ where the response associated with $\boldsymbol{x}_*$ is random but $\mathbb{D}$ is not random is given by

$$MSE(\boldsymbol{x}_*) = \sigma^2 + (\text{Bias}[\boldsymbol{x}_*])^2$$

Here, $\sigma^2 = \text{Var}[\Delta \mid \boldsymbol{x}]$ for all $\boldsymbol{x}$ by one of our assumptions (homoskedasticity). Δ is a random variable from which the ignorance errors are realized. Also, the bias is given by

$$\text{Bias}[\boldsymbol{x}_*] = f(\boldsymbol{x}_*) - g(\boldsymbol{x}_*)$$

where f is the conditional expectation function for the phenomenon in question, and g is a function from some candidate set $\mathcal{H}$ output by an algorithm attempting to fit f. In fact, $\text{Bias}[\boldsymbol{x}_*] := \mathbb{E}[E_* \mid \boldsymbol{x}_*]$ and $MSE[\boldsymbol{x}_*] = \mathbb{E}[E_*^2 \mid \boldsymbol{x}_*]$, but using our assumptions these are shown to be the expressions presented earlier. E_* is a random variable from which residuals are realized, defined as $E_* := Y_* - g(\boldsymbol{x}_*)$.

(b) [easy] Write down (do not derive) the decomposition of MSE for a given $\boldsymbol{x}_*$ where the responses in $\mathbb{D}$ is random but the $\boldsymbol{X}$ matrix is assumed fixed and the response associated with $\boldsymbol{x}_*$ is assumed random like previously.

This time, we have

$$\text{MSE}[\boldsymbol{x}_*] = \sigma^2 + (\text{Bias}[G(\boldsymbol{x}_*)])^2 + \text{Var}[G(\boldsymbol{x}_*)]$$

Here, $G = \mathcal{A}(\mathbb{D}, \mathcal{H})$, and it is a random variable because the responses are random. That is, the randomness in $\Delta_1, \ldots, \Delta_n$ make it so that we have random $Y_1, \ldots, Y_n$, and in turn our prediction function G is derived from a random G.

(c) [easy] Write down (do not derive) the decomposition of MSE for general predictions of a phenomenon where all quantities are considered random.

$$\text{MSE} := \mathbb{E}[\text{MSE}[\boldsymbol{x}_*]] = \sigma^2 + \mathbb{E}_X[(\text{Bias}[G(\boldsymbol{x}_*)])^2] + \mathbb{E}_X[\text{Var}[G(\boldsymbol{x}_*)]]$$

(d) [difficult] Why is it in (a) there is only a "bias" but no "variance" term? Why did the additional source of randomness in (b) spawn the variance term, a new source of error?

Recall the MSE is defined as

$$\text{MSE}[\boldsymbol{x}_*] = \mathbb{E}[(Y_* - g(\boldsymbol{x}_*))^2]$$

In (a), since none of the data in $\mathbb{D}$ is random, the prediction function g is deterministic (i.e., not random) as a function of $\boldsymbol{x}_*$, and similarly f is also. Therefore, in the computation of the MSE as defined above, g comes out of the expectation, and no variance term survives. Put another way, since there is no randomness in any component of $\mathbb{D}$ (i.e. neither the responses nor the features), the g is the same each time.

In (b), once we allow randomness in the responses, this in turn makes the prediction function nondeterministic, meaning it makes our prediction function g into a random variable G. The randomness in the responses in $\mathbb{D}$ leads to construction of different g's. Therefore for any given point $\boldsymbol{x}_*$, our predictions vary depending on what realization of G that we use, and this variability is captured by the variance term.

(e) [harder] A high bias / low variance algorithm is underfit or overfit?

Bias is a measure of underfitting or misspecification error, and variance is a measure of overfitting or estimation error. Since bias is high, underfit errror is high, and since variance is low, overfit error is low. Hence, overall we get underfitting.

(f) [harder] A low bias / high variance algorithm is underfit or overfit?

Similar to before, low bias means there is little underfitting error, and high variance means large estimation error (and hence high overfitting error) (for example we are using junk features). Overall, we have a high overfitting error.

(g) [harder] Explain why bagging reduces MSE for "free" regardless of the algorithm employed.

When bagging, we ship as our prediction function the average of M prediction functions

$$g_{\text{avg}} := \frac{1}{M}\sum_{m=1}^{M} g_m$$

Bagging is done such that the g_m become somewhat decoupled, and this leads to a reduction in the covariance among them. In turn, the covariance reduces the variance term in the MSE, and hence the MSE also reduces.

(h) [harder] Explain why RF reduces MSE atop bagging M trees and specifically mention the target that it attacks in the MSE decomposition formula and why it's able to reduce that target.

Random forests use a subset of the features when building trees rather than all of the features. Therefore the models are built based on different subsets of features, further decoupling them. The result is a reduction in the covariance, and hence of the variance term and overall MSE.

(i) [difficult] When can RF lose to bagging M trees? Hint: think hyperparameter choice.

It can lose for poor choices of the m_{try} hyperparameter. For example, if $m_{\text{try}} = 1$, then each tree built is likely to underfit heavily. As m_{try} increases, the number of possible subsets of features increases, so there are more options for the different trees, while allowing enough degrees of freedom for a better fit.

Problem 3

These are some questions related to missingness.

(a) [easy] [MA] What are the three missing data mechanisms? Provide an example when each occurs (i.e., a real world situation). We didn't really cover this in class so I'm making it a MA question only. This concept will NOT be on the exam.

(b) [easy] Why is listwise-deletion a *terrible* idea to employ in your $\mathbb{D}$ when doing supervised learning?

You risk suffering from sampling bias. For example if your data set has an age feature, and the observations where age is larger (older) shows a lot of missingness, dropping them means that that population is not represented in the model. When we go off to predict, our model will extrapolate poorly.

(c) [easy] Why is it good practice to augment $\mathbb{D}$ to include missingness dummies? In other words, why would this increase oos predictive accuracy?

When some features are missing for a given observation, that missingness may be related to the missingness of other features. Moreover, it's possible that the missingness itself tells us important information about the individual. For example, for a student that is absent from school a lot, the lack of academic records may be as the actual grades they would have had if the grade were present. Thus the model is better equipped to predict out-of-sample.

(d) [easy] To impute missing values in $\mathbb{D}$, what is a good default strategy and why?

A good default strategy is to assume that data is not missing at random and to construct a matrix M of the missing values. That is, the columns of M correspond to columns of X that exhibit missingness. If X is $n \times p$, then M is $n \times q$ for some $q \leq p$. Entry M_{ij} is 1 is the ith unit has missingness for some feature in X index j (not the jth feature of X, but the jth feature missing in X). Then, we compute an imputed matrix by applying an imputation algorithm on the augmented matrix:

$$[X_{\text{imp}} \mid M] = \text{Impute}([X \mid M], \boldsymbol{y})$$

The reason it is a good idea is that data is likely not missing at random; we simply do not know why it is missing. Moreover, the missingness may tell us something about the units themselves, so this can lead to better predictive value.

Problem 4

These are some questions related to gradient boosting. The final gradient boosted model after M iterations is denoted G_M which can be written in a number of equivalent ways (see below). The g_t's denote constituent models and the G_t's denote partial sums of the constituent models up to iteration number t. The constituent models are "steps in functional steps" which have a step size η and a direction component denoted $\tilde{g}_t$. The directional component is the base learner $\mathcal{A}$ fit to the negative gradient of the objective function L which measures how close the current predictions are to the real values of the responses:

$$\begin{aligned} G_M &= G_{M-1} + g_M \\ &= g_0 + g_1 + \ldots + g_M \end{aligned}$$

$$\begin{aligned}
&= g_0 + \eta \tilde{g}_1 + \ldots + \eta \tilde{g}_M \\
&= g_0 + \eta \mathcal{A}\left(\langle \boldsymbol{X}, -\nabla L(\boldsymbol{y}, \hat{\boldsymbol{y}}_1)\rangle, \mathcal{H}\right) + \ldots + \eta \mathcal{A}\left(\langle \boldsymbol{X}, -\nabla L(\boldsymbol{y}, \hat{\boldsymbol{y}}_M)\rangle, \mathcal{H}\right) \\
&= g_0 + \eta \mathcal{A}\left(\langle \boldsymbol{X}, -\nabla L(\boldsymbol{y}, g_1(\boldsymbol{X}))\rangle, \mathcal{H}\right) + \ldots + \eta \mathcal{A}\left(\langle \boldsymbol{X}, -\nabla L(\boldsymbol{y}, g_M(\boldsymbol{X}))\rangle, \mathcal{H}\right)
\end{aligned}$$

(a) [easy] From a perspective of only multivariable calculus, explain gradient descent and why it's a good idea to find the minimum inputs for an objective function L (in English).

Gradient descent is used to determine the inputs on which a function achieves its extremum values. In our context, we care about inputs where it achieves the minimum. Gradient descent achieves this by using the fact that the gradient gives the direction of greatest increase of a scalar-valued function of (possibly several) variables. Thus, the negative gradient gives the direction of greatest decrease, and we expect this will lead towards the minimum. Objective functions generally represent a practical quantity of interest. For example, it may represent the amount of material available to build an object, or the amount of money that can be spent on resources, or the amount of error that can be incurred by a numerical procedure. Thus finding the that minimize these quantities allow us to optimally solve certain problems while respecting constraints.

(b) [easy] Write the mathematical steps of gradient boosting for supervised learning below. Use L for the objective function to keep the procedure general. Use notation found in the problem header.

Let $\mathbb{D} = \langle X, \boldsymbol{y}\rangle$ be a given data set. Fix an algorithm $\mathcal{A}$, a hypothesis set $\mathcal{H}$, a learning rate η, and maximum number of steps M. Let g_0 be a default starting function (for example the null model).

For $t = 1, 2, \ldots, M$:

- Compute $\hat{\boldsymbol{y}}_t = g_{t-1}(X)$.
- Compute $-\nabla L(\boldsymbol{y}, \hat{\boldsymbol{y}}_{t-1})$.
- Compute a prediction function

$$\tilde{g}_t := \mathcal{A}(\langle X, -\nabla L(\boldsymbol{y}, \hat{\boldsymbol{y}}_t)\rangle, \mathcal{H})$$

- Set $G_t := g_{t-1} + \eta \cdot \tilde{g}_t$.

(c) [easy] For regression, what is $g_0(\boldsymbol{x})$?

$g_0(\boldsymbol{x}) = \text{Mean}[\boldsymbol{y}]$.

(d) [easy] For probability estimation for binary response, what is $g_0(\boldsymbol{x})$?

$g_0(\boldsymbol{x}) =$ Proportion of 1's in $\boldsymbol{y}$.

(e) [harder] What are all the hyperparameters of gradient boosting? There are more than just two.

- M, the number of models to use.
- η, the learning rate.
- g_0, the default model for the 0th iteration.
- L, the loss function.

(f) [easy] For regression, rederive the negative gradient of the objective function L.

In this setting, we can use the SSE as the loss function:

$$L(\boldsymbol{y}, \hat{\boldsymbol{y}}) = SSE := \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

Then the negative gradient is computed by taking partial derivatives with respect to $\hat{y}_i$:

$$-\nabla L(\boldsymbol{y}, \hat{\boldsymbol{y}}) = -\begin{bmatrix} \frac{\partial L}{\partial \hat{y}_1} \\ \vdots \\ \frac{\partial L}{\partial \hat{y}_n} \end{bmatrix} = -\begin{bmatrix} -(y_1 - \hat{y}_1) \\ \vdots \\ -(y_n - \hat{y}_n) \end{bmatrix} = 2(\boldsymbol{y} - \hat{\boldsymbol{y}}) = 2\boldsymbol{e}$$

(g) [easy] For probability estimation for binary response, rederive the negative gradient of the objective function L.

We begin with the probability estimation function (assuming a logistic link function):

$$P(\boldsymbol{y}, \hat{\boldsymbol{p}}) = \prod_{i=1}^{n} \hat{p}_i^{y_i} (1 - \hat{p}_i)^{1-y_i}$$

where we have

$$\hat{p}_i = \phi(\hat{y}_i) = \frac{e^{\hat{y}_i}}{1 + e^{\hat{y}_i}} \implies \hat{y}_i = \ln\left(\frac{\hat{p}_i}{1 - \hat{p}_i}\right)$$

We write P as a function of $\boldsymbol{y}$ and $\hat{\boldsymbol{p}}$:

$$\begin{aligned} P(\boldsymbol{y}, \hat{\boldsymbol{p}}) &= \prod_{i=1}^{n} \hat{p}_i^{y_i} (1 - \hat{p}_i)^{1-y_i} \\ &= \prod_{i=1}^{n} \left(\frac{e^{\hat{y}_i}}{1 + e^{\hat{y}_i}}\right)^{y_i} \cdot \left(\frac{1}{1 + e^{\hat{y}_i}}\right)^{1-y_i} && \text{(Substitute expression for } \hat{p}_i\text{)} \end{aligned}$$

$$\begin{aligned} &= \prod_{i=1}^{n} \frac{e^{y_i \hat{y}_i}}{1+e^{\hat{y}_i}} \\ &= P(\boldsymbol{y}, \hat{\boldsymbol{y}}) \end{aligned}$$

Then we find the loss function L in as the logarithm of the probability:

$$\begin{aligned} L(\boldsymbol{y}, \hat{\boldsymbol{y}}) &= \ln(P(\boldsymbol{y}, \hat{\boldsymbol{y}})) \\ &= \ln\left(\prod_{i=1}^{n} \frac{e^{y_i \hat{y}_i}}{1+e^{\hat{y}_i}}\right) \\ &= \sum_{i=1}^{n} \ln\left(\frac{e^{y_i \hat{y}_i}}{1+e^{\hat{y}_i}}\right) && \text{(Product property of logarithms)} \\ &= \sum_{i=1}^{n} (y_i \hat{y}_i - \ln(1+e^{\hat{y}_i})) && \text{(Quotient property of logarithms)} \end{aligned}$$

Finally, we compute the gradient:

$$-\nabla L(\boldsymbol{y}, \hat{\boldsymbol{y}}) = -\begin{bmatrix} \frac{\partial L}{\partial \hat{y}_1} \\ \vdots \\ \frac{\partial L}{\partial \hat{y}_n} \end{bmatrix} = -\begin{bmatrix} y_1 - \frac{e^{\hat{y}_1}}{1+e^{\hat{y}_1}} \\ \vdots \\ y_n - \frac{e^{\hat{y}_n}}{1+e^{\hat{y}_n}} \end{bmatrix} = \boldsymbol{y} - \hat{\boldsymbol{p}} = \boldsymbol{e}$$

(h) [difficult] For probability estimation for binary response scenarios, what is the unit of the output $G_M(\boldsymbol{x}_\star)$?

Probability. Note that the unit of output of G_M is precisely the unit of output of g_0, because G_M is a sum of functions, and g_0 is one of the terms in that sum. For gradient boosting, a valid g_0 is the proportion of 1's in the response $\boldsymbol{y}$. This proportion is therefore used as a probability threshold for deciding whether to predict 1 for a unit.

Another way to argue this is to note that the negative gradient of the loss function is $\boldsymbol{y} - \hat{\boldsymbol{p}}$. The quantity $\boldsymbol{y}$ is unitless but $\hat{\boldsymbol{p}}$ has units of probability. Their difference has units of probability.

(i) [easy] For the base learner algorithm $\mathcal{A}$, why is it a good idea to use shallow CART (which is the recommended default)?

Shallow trees have low variance, which tends to be a big component of the MSE. Even though they have high bias, the boosting approach iteratively reduces bias by accruing several weak learners, while maintaining covariance low.

(j) [difficult] For the base learner algorithm $\mathcal{A}$, why is it a bad idea to use deep CART?

Deep CART are strong learners; they overfit. Since variance is a measure of estimation and overfitting error, this means they are high variance models. Since they overfit, they begin with high bias. The algorithm works by trying to minimize loss, which is tantamount to reducing bias. But there is no much bias to reduce because the learner is overfitting, so the prediction functions produced by each iteration will not change much from the previous iteration. Hence, the covariance will not decrease by much, even though the bias is low, and we will not reduce the overall significantly MSE as intended.

(k) [difficult] For the base learner algorithm $\mathcal{A}$, why is it a bad idea to use OLS for regression (or logistic regression for probability estimation for binary response)?

Intuitively, if the data does not admit a linear trend and we use linear models as learners, then their sum will still be linear.

More generally, OLS is not suitable exhibiting nonlinearities. Unless we introduce transformations that help capture the essence of the phenomenon, the result will be models that severely underfit, and the sum obtained from the boosting approach will inherit this issue.

(l) [difficult] If M is very, very large, what is the risk in using gradient boosting even using shallow CART as the base learner (the recommended default)?

You risk overfitting. Each iteration, we try to reduce the residual errors. After too many iterations, we may start fitting noise (i.e. ignorance error).

(m) [difficult] If η is very, very large but M reasonably correctly chosen, what is the risk in using gradient boosting even using shallow CART as the base learner (the recommended default)?

In gradient *descent*, an η that is too large can lead to missing the minimum value we seek. Analogously, a very large η may mean that the G_t does not result in a meaningful reduction in the residual, what we are trying to minimize across iterations.