Lecture 23: MATH 342W: Introduction to Data Science and Machine Learning

Sergio E. Garcia Tapia[*]

May 6th, 2025 (last updated May 9, 2025)

1 Bagging for Classification

Recall that in the context of regression, we defined g_{avg} by

$$g_{\text{avg}} := \frac{g_1 + g_2 + \cdots + g_M}{M}$$

where g_i are fit on different bootstrap samples to get more independence and drive down the variance term in the Bias-Variance Trade-off. If $\mathcal{Y}$ is categorical, then we define g_{avg} using the mode:

$$g_{\text{mode}} := \text{Mode}[g_1, g_2, \ldots, g_M]$$

We did not talk about the Bias-Variance Trade-off in this context, but there is analogous theory, though more complicated and we will not go into it here.

2 Missingness in Future Units

Last time we talked about the need to perform imputation to deal with missingness in our data. Let's consider again what happens when we do a train-test split in the presence of missingness, as in Figure 1. We want to construct binary matrix M whose entries indicate missingness of entries in X, and then impute with the MissForest algorithm that

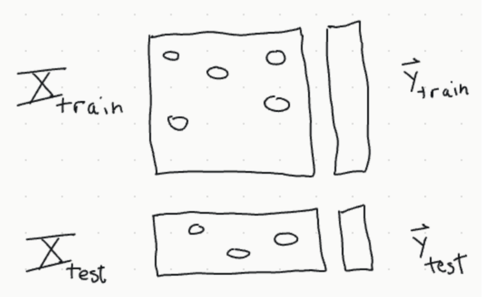

Figure 1: Train-test split of a data set $\mathbb{D}$ that exhibits missingness.

[*] Based on lectures of Dr. Adam Kapelner at Queens College. See also the course GitHub page.

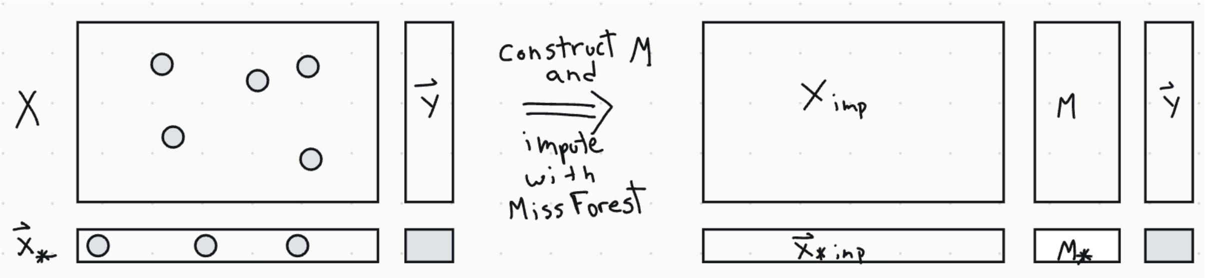

Figure 2: Predicting on a future unit that has missingness by recomputing the entire model.

we saw. However, we want to be careful not to look at $\boldsymbol{y}_{\text{test}}$ because it may influence our g, thereby increasing the risk that we are dishonest. Therefore, we make $\boldsymbol{y}_{\text{test}}$ entries all missing (in R, this means `NA`), and then we impute. The algorithm imputes the missing values for both X_{test} and $\boldsymbol{y}_{\text{test}}$, and we can discard the latter because not only do we already have them (we artificially made them missing for correctness), but these are the values we are predicting to begin with (the $\hat{\boldsymbol{y}}_{\text{test}}$ we will compute).

At the end of this, we will have computed a prediction function g:

$$g = \mathcal{A}(\langle [X_{\text{imp}}, M], \boldsymbol{y} \rangle, \mathcal{H})$$

So far, all of this has been review to tackle the following question: how do we predict in the future for a unit $\boldsymbol{x}_*$ exhibiting missingness? The short answer is that you cannot, and we must fill in the missingness. There are three approaches we can try:

(1) **Recompute (slowest but most accurate)**: This means we impute X_{updated}, consisting of the n rows of X and $\boldsymbol{x}_*$ as row $n+1$. Then we create a new predictive function:

$$g_{\text{updated}} = \mathcal{A}(\langle [X_{\text{imp, updated}}, M_{\text{updated}}], \boldsymbol{y} \rangle, \mathcal{H})$$

See Figure 2. Because we have more data, this may result in an improvement, but it can be expensive. This may be reasonable if we are doing batch predicting, meaning that we are predicting for much more than just one new unit.

(2) **Reuse (Fastest but least accurate)**: In this case, we do not start the imputation from our original X. We start with $[X_{\text{imp}} \mid M]$, and we impute on the matrix with $n+1$ rows consisting of $[X_{\text{imp}} \mid M]$ and $[\boldsymbol{x}_* \mid M_*]$. Because there is only one row of missing data now, it is much faster. After imputing, thereby eliminating the missingness in $\boldsymbol{x}_*$, we use the old g to predict:

$$\hat{\boldsymbol{y}} = g([\boldsymbol{x}_{*,\text{imp}} \mid M_*])$$

See Figure 3.

(3) **Middle strategy**: Do the imputation from (1), but do not refit g. Use the old g to predict.

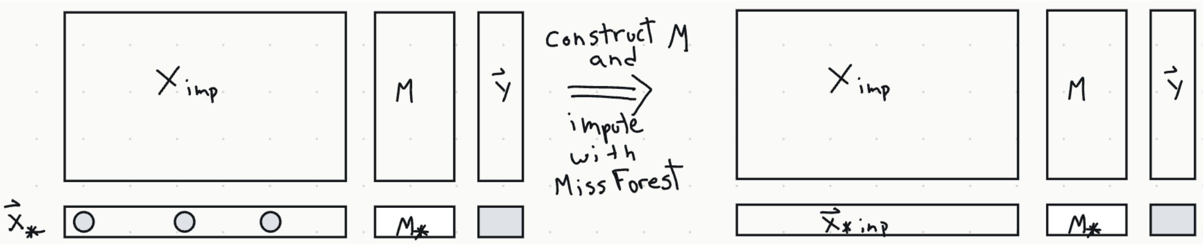

Figure 3: Predicting on a future unit that has missingness by reusing the existing model.

3 Probability Estimation in Asymmetric Cost Modeling

We saw that in a classification setting, we may need to distinguish between false positives and false negatives; they both correspond to mispredictions, but one may be more costly than the other. If C_{FP} is the cost of false-positives, C_{FN} is the cost of false negatives, and $C_{FP} > C_{FN}$ (say, $C_{FP} = 100 \cdot C_{FN}$), then we want to be absolutely sure that when we predict $\hat{y} = 1$, the it is indeed a true positive. How do incorporate this heuristic in our model?

For example, instead of using the mode (highest frequency), can we do something else? The canonical way to do this is with *probability estimation.* We seek something of the form

$$\hat{p}_* = g(\boldsymbol{x}_*) \implies \hat{y}_* = \underset{\hat{p}_* \geq p_{\text{th}}}{\mathbb{I}}$$

In other words, we predict the probability that a new unit $\boldsymbol{x}_*$ corresponds to a true positive. If that probability exceeds a threshold p_{th}, then we predict positive for $\boldsymbol{x}_*$. The value p_{th} is called the **probability threshold** and it is a hyperparameter. For example, it may be 50%. We want to be very sure that when we say $\hat{y} = 1$, it is indeed the case that $y = 1$ since C_{FP} is so high. As C_{FP} increases, we may want to increase p_{th} even more, perhaps all the way to $p_{\text{th}} \geq 99\%$.

To fit asymmetric cost classificaiton models using an underlying probability estimation model, define a grid of p_{th} values, for example:

$$p_{\text{th, grid}} = \{0.001, 0.002, \ldots, \ldots, 0.998, 0.999\}$$

Then compute a confusion matrix and performance metrics for each value of p_{th}[1]:

p_{th}	TP	TN	FP	FN	Precision	Recall	FPR	FDR	FOR	C
0.001										
0.002										
$\vdots$										
0.999										

Then, find p_{th} where C is minimized. The procedure just described is the standard, but there are alternatives. In one alternative, we take the values C_{FP} and C_{FN} to be negative

[1] Notice that changing p_{th} changes the values of the confusion matrix (that is, TN, FN, FP, TP). However, N and P do not change because these are the observed responses, so changing p_{th} does not change how many y's are 0 or 1, or which values are 0 or 1. Similarly, n does not change.

(unlike the procedure described, where they are positive), and instead of *minimizing* cost C, we *maximize* **winnings** W:

$$W := \underbrace{W_{\text{TP}}}_{>0} \cdot TP + \underbrace{W_{\text{TN}}}_{>0} \cdot TN + \underbrace{C_{FP}}_{<0} \cdot FP + \underbrace{C_{FN}}_{<0} \cdot FN$$

Winnings give us a more complete picture. However in this course we will focus on minimizing C.

Let's consider the effect on increasing p_{th} as we step through the grid. As p_{th} increases, it becomes harder to predict $\hat{y} = 1$ because the probability threshold is higher. As a result, the number of false positives (FP) and true positives (TP) will decrease, and hence the number of false negatives (FN) and true negatives (TN) will increase. Remember that P and N stay the same since the response is not changing throughout this process. The effect on the some of the remaining metrics is as follows:

- **Recall**: Defined as $\frac{TP}{P}$, this value will decrease because P remains constant and TP is decreasing as p_{th} increases.

- **Precision**: Defined as $\frac{TP}{PP}$, this value increases, Note that since both FP and TP increase, we see an increase in PP. Thus, both the numerator TP and the denominator PP decrease, making it harder to immediately see how this quantity changes. However, note that the model is more sure of its predictions as p_{th} increases. Therefore, the ratio of TP to PP will increase.

- **False Discovery Rate (FDR)**: Defined as $\frac{FP}{PP} = 1 - \text{Precision}$, this value decreases.

- **FOR**: Defined as $\frac{FN}{PN}$, this value increases. This is similar to the reason for an increase in Precision; the number of predicted negatives (PN) increases because the model is attempting to play it safe, but we incur a higher number of false negatives (FN) also.

3.1 More Metrics for Asymmetric Cost Modeling

Other metrics sometimes come into play, or different names for metrics we have seen. Among them we define **sensitivity**, defined as

$$\text{Sensitivity} := \text{Recall}$$

and **specificity**:

$$\text{Specificity} := \frac{TN}{N} = 1 - \frac{FP}{N} = 1 - FPR$$

Incidentally, FPR is the **false positive rate**, defined as

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

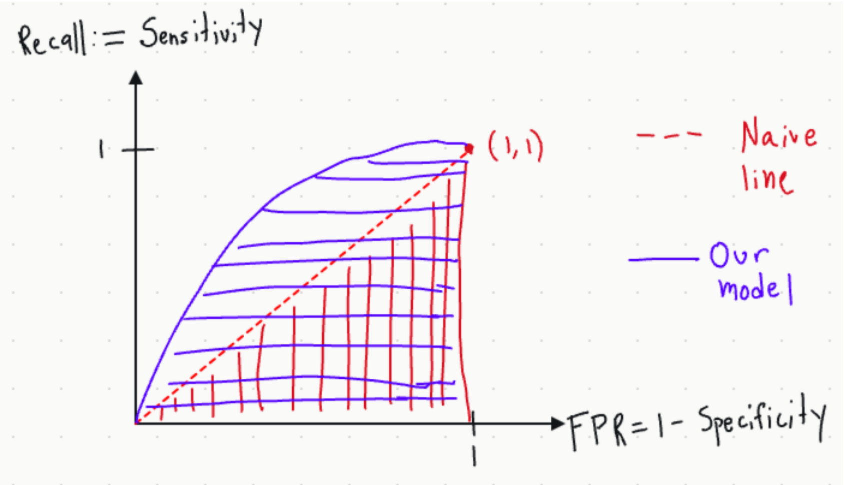

Figure 4: A plot of Recall vs. FPR. The dotted line is the naive model, and the solid line is "our model". Each point on each curve corresponds to a distinct value of p_{th}, since both FPR and Recall are functions of p_{th}.

4 Receiver Operator Curve

Consider a naive probability estimation model

$$g_{\text{naive}}(\boldsymbol{x}_*) = \text{ produces a realization from } U(0,1)$$

where $U(0,1)$ is the uniform distribution on the interval $(0,1)$. This is akin to flipping a coin, and is different from g_0, which predicts according to the proportion of 1's in the response. In fact, g_{naive} performs *worse* than g_0. We would like to beat g_{naive}, so in the interest of comparing a model against it, let's look at its confusion matrix. The predicted values for g_{naive}, the $\hat{p}$ values generated from $U(0,1)$, can be used to compute the predictions $\hat{y} = \underset{\hat{p}\geq p_{\text{th}}}{\mathbb{I}}$. The confusion matrix becomes:

		$\hat{\boldsymbol{y}}$		
	0	$TN = p_{\text{th}} \cdot N$	$FP = (1 - p_{\text{th}}) \cdot N$	N
$\boldsymbol{y}$	1	$FN = p_{\text{th}} \cdot P$	$TP = (1 - p_{\text{th}}) \cdot P$	P
		$p_{\text{th}} \cdot PN$	$(1 - p_{\text{th}}) \cdot PP$	n

With these values, the Recall becomes

$$\text{Recall} := \frac{TP}{P} = \frac{(1 - p_{\text{th}}) \cdot P}{P} = 1 - p_{\text{th}}$$

and the FPR is

$$FPR := \frac{FPR}{N} = \frac{(1 - p_{\text{th}}) \cdot N}{N} = 1 - p_{\text{th}}$$

Thus we see that for g_{naive}, Recall equals FPR. We would like to maximize Recall and minimize FPR.

Figure 4 plots Recall vs. FPR for the g_{naive} with a dashed line. Each point corresponds to a different value of p_{th} (as given by our grid). It is a straight line since Recall $= FPR$ for g_{naive}, as we showed. The area under the curve of g_{naive} is 0.5. The figure also shows

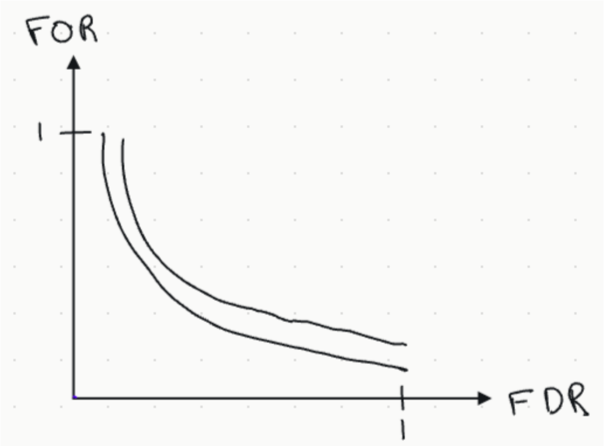

Figure 5: Comparing models by plotting FDR vs. FOR.

a curve for a hypothetical model in a solid line. The area under the curve is larger than that of g_{naive}. We call this curve the **Receiver Operator Curve (ROC)**. The ROC allows for comparisons among probability estimation models, similar to the Brier and log scoring rules. It is computationally intensive, but some prefer it gives a very intuitive comparison. For example, if the area under the curve comes out to 0.501, then the model barely outperforms picking numbers at random.

5 FPR vs. FOR

Let's revisit the definitions of FPR and FOR:

$$FPR := \frac{FP}{PP} \qquad FOR := \frac{FN}{PN}$$

Note that t he other error metrics are a function of N and P, but that's not reality because we do not have access to these (we do not get to see the y's). That is why FDR and FOR are better (this is an opinion). So, this motivates a curve that graphs FOR vs. FOR. See Figure 5 Note the trade-off; high FDR corresponds to low FOR, and viceversa. Unlike the Recall vs. FPR line, there is no comparison against a naive line, but we can use it to compare models. It is useful for reading error rates, which is what we are interested in when we predict in the future.