Variable Importance

Duke Course Notes

Cynthia Rudin

Sources: Fisher et al, (2019), Breiman (2001)

An understanding of variable importance is useful for a multitude of reasons. However, the term "variable importance" doesn't have meaning without context. Variable importance to what entity? People use the term all the time to mean totally different things! And it is confusing.

1. How important is the variable *to a given machine learning model*?
2. How important is the variable's *unique information* (that cannot be expressed by other variables) *to a given machine learning model*?
3. How important is the variable *for a specific prediction given by a specific machine learning model*?
4. How important is the variable *to a given algorithm on a dataset?*
5. How important is the variable *to the dataset*? This last one is too general, so let's make it more specific: How important is the variable *to any good predictive model on the dataset*?
6. How important is the variable's *unique information* (that cannot be expressed by other variables) *to the dataset*? Again, let's make it more specific: How important is the variable's *unique information to any good predictive model on the dataset*?

There are a multitude of answers to any of these questions. We will pick one or two for each.

1. How important is the variable *to a given machine learning model*? When asking this question, you must remember that this doesn't tell you how much any reasonable model would use that variable, just how much a given model uses the variable.

For linear models, one could use the magnitude of the variable's coefficient. To do this, you must remember to normalize the range of the variable to [0,1]. (If

you don't, a variable that takes tiny values and thus has huge coefficients will always appear to be very important!)

For nonlinear models, you can use model reliance, which is a permutation importance technique. Random forests comes with its own variable importance technique. I'll describe both. They have a disadvantage that I'll discuss.

Model Reliance (MR):
This is a simple technique: we scramble the variable (messing up its joint distribution with other variables and with the label), and determine how much the loss increased when we did it. To do the scrambling, we would randomly permute the jth variable (reorder the jth column) of the matrix $\mathbf{X}$, where each feature is a column. I will denote the matrix $\mathbf{X}$ with the jth column scrambled as $\mathbf{X}^{j \text{ scrambled}}$. The model reliance of model f on variable j is:

$$MR_{\text{diff}}(f, j) := \text{Loss}(f(\mathbf{X}^{j \text{ scrambled}}), \mathbf{Y}) - \text{Loss}(f(\mathbf{X}), \mathbf{Y}).$$

(Here I used matrix notation, where Loss is the empirical risk, that is, the sum of losses over all i.) If $MR_{\text{diff}}(f, j) = 0$, it means the variable wasn't important, because the loss didn't change when we messed up that variable.

Why do we scramble the variable rather than setting it to 0? Sometimes it doesn't matter, but sometimes it does: if your model is a decision tree, then setting a variable to 0 could send all of the observations down one branch of the tree that has a split on that variable, which you don't want. When you scramble the variable, you are destroying its joint distribution with the other features and the label, but you are preserving its marginal distribution.

We can instead define MR as a ratio rather than a difference:

$$MR_{\text{ratio}}(f, j) := \frac{\text{Loss}(f(\mathbf{X}^{j \text{ scrambled}}), \mathbf{Y})}{\text{Loss}(f(\mathbf{X}), \mathbf{Y})}.$$

The values from $MR_{\text{ratio}}(f, j)$ have a natural interpretation. If $MR_{\text{ratio}}(f, j)$=1, then the variable wasn't important, since the scrambled error is the same as the regular error. If $MR_{\text{ratio}}(f, j)$=2, it means the error doubled when we scrambled the variable. If $MR_{\text{ratio}}(f, j)$=1.5, the error went up by half its usual amount when we scrambled the variable.

There are actually a few different variations of how to scramble the variable and calculating the loss. For instance, instead of randomly permuting the values of j, we could use all pairwise swaps. For ease of notation, let us say the variable of interest is variable 1. I can swap point i's value of variable 1 with point k's value, for each possible pair of points i and k.

$$\frac{1}{n(n-1)} \sum_{i,k:i \neq k} \text{loss}(f([x_{i,1}, \mathbf{x}_{i,2,..p}]), y_i).$$

If that is too computationally expensive, I could just take the value of variable 1 from the first half of the dataset, and swap it with variable 1's values for the second half of the dataset, and add up the losses for these n newly formed data points.

Model reliance does have a disadvantage, which is that when you use predictions from a scrambled variable, your scrambled data points may be outside the distribution that the data typically reside (Hooker and Mentch, 2019). For instance, you could have a datapoint representing a 40-year-old married male with diabetes, and when you scramble age, it could become a 2-year-old married male with diabetes. The model might behave in unexpected ways on this new data point because it wasn't trained on data similar to this datapoint. That said, I still think model reliance is useful, for the simple reason that if the variable does not appear in the model, the model reliance will be around 0 (or 1 in the case of MR_{ratio}), because you generally won't predict better out-of-distribution than you could predict within-distribution.

Random Forest's Variable Importance:
Random forest's variable importance is essentially model reliance calculated on data that wasn't used in each tree. (Remember, when you take a bootstrap sample, it would typically include some data points twice and some not at all.) So it uses out-of-sample data for each tree t that was built in the forest. Here is how it works:

- Take the data not used to construct tree t. Call it out-of-bag data for tree t, OOB_t.
- Compute $\text{loss}_t(\text{tree}_t, \text{OOB}_t)$, the loss for model tree_t on data OOB_t.

- Now, randomly permute variable j, call this $\text{OOB}_{t,\text{scrambled}}$. Compute loss($\text{tree}_t$, $\text{OOB}_{t,\text{scrambled}}$).
- Calculate the variable importance for variable j as the average difference in losses between the permuted and original data:

$$\frac{1}{T}\sum_t \left[\text{Loss}_t(\text{tree}_t, \text{OOB}_{t,\text{scrambled}}) - \text{Loss}_t(\text{tree}_t, \text{OOB}_t)\right].$$

This is nice in that it is essentially an out-of-sample calculation within a training set.

Back to the questions we could ask about variable importance:

2. How important is the variable's unique information (that cannot be expressed by other variables) *to a given machine learning model*?

Here, we'll describe conditional model reliance.

Conditional Model Reliance. If features are highly correlated, a model might choose one of them (call it variable 1). However, we want to know how much the model relies on information from $x_{\cdot,1}$ that cannot be gleaned from the rest of the variables, which we call $\mathbf{x}_{\cdot,2\ldots p}$. (I'm using notation $x_{\cdot,1}$ to mean that I am looking at variable 1, without choosing a particular observation, which would have been denoted $x_{i,1}$.)

To get the answer to this, we try to scramble just the part of $x_{\cdot,1}$ that cannot be predicted using $\mathbf{x}_{\cdot,2\ldots p}$.

Step 1. For each observation i, we're going to predict its value of $x_{i,1}$ from its value of $\mathbf{x}_{i,2\ldots p}$. I'll denote that with $\hat{x}_{i,1}$. We could write that $\hat{x}_{i,1}$ is an estimate of the expectation of the random variable $X_{i,1}$ when variables $2\ldots p$ are set to $\mathbf{x}_{i,2\ldots p}$:

$$\hat{x}_{i,1} = \hat{\mathbb{E}}(X_{i,1}|X_{i,2\ldots p} = \mathbf{x}_{i,2\ldots p}).$$

A tricky part about this calculation is that you need a separate machine learning algorithm to make this prediction, and it is not clear which one to use. Probably I would use kernel least squares (which is explained later) if the variable is

continuous, or logistic regression or boosting (both explained later) for binary variables. We can think about $\hat{x}_{i,1}$ as containing the information about $x_{i,1}$ that can be gleaned from other variables. The "unique" information to variable 1 for point i is then what remains:

$$x_{i,1} - \hat{x}_{i,1}.$$

Step 2. Now we'll scramble the unique information. We will take the unique information from point i and give it to a different point k, whose unique information has been removed.

$$x_{ik,1}^{\text{scrambled}} = [x_{i,1} - \hat{x}_{i,1}] + \hat{x}_{k,1}.$$

Now, we have successfully created a data point where the unique information has been scrambled. We'll put this new value for the first variable alongside the rest of k's variables:

$$\mathbf{x}_{ik}^{\text{scrambled}} = [x_{ik,1}^{\text{scrambled}}, \mathbf{x}_{k,2\ldots p}].$$

This point is actually $\mathbf{x}_k$ but where k's unique information about variable 1 was replaced with i's unique information about variable 1. I like to think of this new vector $\mathbf{x}_{ik}^{\text{scrambled}}$ as a sort of chimera, with the body of point k and the head of point i.

Step 3. Calculate the scrambled loss for all of these "chimeric" points:

$$\text{ScrambledLoss}_1 = \frac{1}{n(n-1)} \sum_{ik:i \neq k} \text{loss}(y_k, f(\mathbf{x}_{ik}^{\text{scrambled}})).$$

And then, as usual, you could use a difference or ratio to compare the scrambled loss to the usual loss. I'll use the ratio to compute conditional model reliance:

$$CMR_{\text{ratio}}(f, 1) := \frac{\text{ScrambledLoss}_1}{\text{Loss}(f(\mathbf{X}), \mathbf{Y})}.$$

Note that using *CMR* assumes homogeneous residuals, which is a weaker assumption than we implicitly assumed for *MR*. That is, *CMR* would be less likely to consider predictions for points that are far outside the distribution of the data.

If the conditional model reliance (as defined above) is 1.1, then the unique information that variable 1 provides would lead to a 10% increase in the loss if it were damaged. If the CMR value is 1, then there is no unique information in variable

1; we can glean all the information we need about variable 1 from other variables.

3. How important is the variable *for a specific prediction given by a specific machine learning model*?

For linear models, $f(\mathbf{x}_i) = \mathbf{b}^T\mathbf{x}_i = b_1x_{i,1} + b_2x_{i,2} + ... + b_jx_{i,j} +, ...$. So for variable j, its contribution to the prediction is $b_jx_{i,j}$.

When we get to generalized additive models (GAMs), they will also tell you exactly how much a variable contributed to each prediction.

For nonlinear models that are not inherently interpretable, it becomes more difficult. There is a whole cottage industry of methods that do this, though that goes beyond today's lecture.

4. How important is the variable *to a given algorithm on a dataset?*

This question is much easier to answer than some of the earlier questions. Here, we run the algorithm with and without the variable and see how much the loss changes. Let us denote this as *algorithm reliance* (AR):

$$AR(\text{variable } j, \text{algorithm } A) = \frac{\text{Loss of algorithm } A \text{ using all variables except } j}{\text{Loss of algorithm } A \text{ using all variables}}.$$

Here, if the algorithm actually needed the variable to perform well, AR would be large. If the algorithm could perform equally well with and without the variable, its value would be 1.

I use algorithmic reliance quite a lot. It is really useful because it tells you how much the algorithm needs to rely on a variable. It tells you about *necessity* of the variable.

(Note that it does not tell you how much a single model f uses a variable since AR uses multiple models. If you are studying a single model, *MR* or *CMR* would be relevant.)

5. How important is the variable *to the dataset*? This last one is too general, so let's make it more specific: How important is the variable *to any good predictive*

model on the dataset?

This question interests a lot of people. Unfortunately people who often want the answer to this question mistakenly end up calculating the answer to other problems instead because they do not understand the differences between these variable importance questions. Unfortunately it means they will get the wrong answer if they don't ask the right question!

What we would like is the *most* and the *least* that *any* good predictive model will depend on variable j. The problem with this question is that we cannot actually enumerate or optimize over *all good* predictive models. Thus, to make the problem computable, we will optimize over all good models within a chosen function class $\mathcal{F}$. This is called *Model Class Reliance*.

Model Class Reliance. Let us define the set of all good models in class $\mathcal{F}$ to be the set of models with loss that is not much more that of a reference model f_{ref} that is known to be reasonably good. We call this the *Rashomon Set*.

$$Rset(\mathcal{F}, \epsilon, f_{\text{ref}}) = \{f \in \mathcal{F} : \text{Loss}(f(\mathbf{X}), Y) \leq \text{Loss}(f_{\text{ref}}, Y) + \epsilon\}\,.$$

Then we compute the range of possible model reliance values over functions in the Rashomon set.

$$[MCR_{-}(\epsilon, j), MCR_{+}(\epsilon, j)] := \left[\min_{f \in Rset(\mathcal{F}, \epsilon, f_{\text{ref}})} MR(f, j), \max_{f \in Rset(\mathcal{F}, \epsilon, f_{\text{ref}})} MR(f, j)\right].$$

This range (these two numbers) constitutes the model class reliance.

The name *Rashomon* comes from a Japanese movie, and it was used by Leo Breiman (Breiman, Two Cities) to describe the notion that there may not be a "best" model, but instead there are many almost-equally-good models, none of which might represent the "truth."

If the MCR_{+} value is small, it means there are no good models in $\mathcal{F}$ that heavily use variable j. For instance, say we are using the ratio version of model reliance, MR_{ratio}. Then, if the range is $[1, 1.001]$, it means that permuting variable j for any good model would increase error only by at most .1%. In that case, it's safe to remove the variable, since no good models use it.

If the *MCR* range is large, it means that there are some good models that heavily use j and some that do not. For instance, in criminal justice, race variables are usually correlated with age and criminal history (due to systemic racism in society). Thus, if we were to predict future re-offending, we might see an MCR that looks like [1,2] for a race variable. This means that there are some low-loss models that do not depend on race at all, and some models that depend on it a lot. Since race doesn't cause crime, some people might be surprised that it can be predictive. For many applications in criminal justice, financial lending and beyond, we omit explicit race variables, though we should mention that information about race is often contained in correlated variables, as shown by the *MCR*.

If the *MCR* range consists of all large numbers (e.g., [1.4-3]), it means that all good models from $\mathcal{F}$ rely on variable j. For instance, we might find that all models for criminal re-offense heavily rely on criminal history features.

MCR is not trivial to calculate, but it has a direct method of calculation for linear models and models from reproducing kernel Hilbert spaces (which will be discussed later, see Fisher et al, 2019). These are very flexible classes of models, because they can include linear combinations of nonlinear combinations of the original variables.

6. How important is the variable's *unique information* (that cannot be expressed by other variables) *to the dataset*? Again, let's make it more specific: How important is the variable's *unique information to any good predictive model on the dataset*?

Here, we can use *conditional model class reliance (CMCR)*, which is a combination of *CMR* and *CMCR* (Fisher et al, 2019):

$$[CMCR_{-}(\epsilon, j), CMCR_{+}(\epsilon, j)] := \left[\min_{f \in Rset(\mathcal{F}, \epsilon, f_{\text{ref}})} CMR(f, j), \max_{f \in Rset(\mathcal{F}, \epsilon, f_{\text{ref}})} CMR(f, j) \right].$$

Again, this is not trivial to calculate; calculating *CMCR* for different model classes could be an interesting future area of research. *CMCR* is probably closer to what scientists actually want when exploring importance of variables. It has some major advantages: like *CMR*, it looks only at the unique information provided by a variable that cannot be gleaned from other variables; and, like

MCR, it is model independent, depending only on a model class that is hopefully large enough to approximate all good models. Thus, it should reveal general information that is unique to the variable, and thus a property of a dataset rather than of a single model.