Feature Importance

DS 6030 | Fall 2023

feature-imp.pdf

Contents

1 Shapley Values for ML

1.1 Shapley Values in Cooperative Game Theory

Suppose up to $p = 4$ singers can cooperate together to make a song. Let's name the people $\{Alicia, Bob, Cardi, Drake\}$. They got together, made a album, and generated \$100M. How much should each artist receive? Is there a fair way to distribute the profits? Shapley values is one "fair" way to distribute the value (see Wikipedia: Shapley Values for the specifics).

Let $S \subseteq \{A, B, C, D\}$ be a *coalition* of the singers and $V(S)$ be the success/value (e.g., sales) of the songs they make. Pretend we can form all possible $\binom{4}{0} + \binom{4}{1} + \binom{4}{2} + \binom{4}{3} + \binom{4}{4} = 16$ coalitions and can measure the value for each one (note: implicit assumption of independence and non-stochasticity). Let the value of the empty set $V(\emptyset) = 0$.

The *shapley score* for person j is:

$$\phi_j = \frac{1}{\text{num of people}} \sum_{\text{coalitions not involving player } j} \frac{\text{gain when } j \text{ is added to the coalition}}{\text{number of coalitions not involving } j \text{ of this size}}$$

$$= \frac{1}{p} \sum_{S:j \notin S} \binom{p-1}{|S|}^{-1} V(S \cup j) - V(S)$$

$$= \frac{1}{p} \sum_{k=0}^{p-1} \binom{p-1}{k}^{-1} \sum_{S:|S|=k} V(S \cup j) - V(S)$$

For example, suppose the following coalition scores:

Coalition	Value
$\emptyset$:	\$0
Alicia:	\$40
Bob:	\$30
Cardi:	\$20
Drake:	\$10
Alicia and Bob:	\$75
Alicia and Cardi:	\$55
Alicia and Drake:	\$50
Bob and Cardi:	\$50
Bob and Drake:	\$40
Cardi and Drake:	\$25
Alicia, Bob, and Cardi:	\$95
Alicia, Bob, and Drake:	\$80
Alicia, Cardi, and Drake:	\$70
Bob, Cardi, and Drake:	\$60
Alicia, Bob, Cardi, and Drake:	\$100

Alicia	Bob	Cardi	Drake	k	V	wt
0	0	0	0	0	0	0.250
1	0	0	0	1	40	0.083
0	1	0	0	1	30	0.083
0	0	1	0	1	20	0.083
0	0	0	1	1	10	0.083
1	1	0	0	2	75	0.083
1	0	1	0	2	55	0.083
1	0	0	1	2	50	0.083
0	1	1	0	2	50	0.083
0	1	0	1	2	40	0.083
0	0	1	1	2	25	0.083
1	1	1	0	3	95	0.250
1	1	0	1	3	80	0.250
1	0	1	1	3	70	0.250
0	1	1	1	3	60	0.250
1	1	1	1	4	100	Inf

Consider all of Alicia's contributions:

Coalitions	Values	Difference
V(Alicia) - V(∅)	= 40 - 0	= 40
V(Alicia, Bob) - V(Bob)	= 75 - 30	= 45
V(Alicia, Cardi) - V(Cardi)	= 55 - 20	= 35
V(Alicia, Drake) - V(Drake)	= 50 - 10	= 40
V(Alicia, Bob, Cardi) - V(Bob, Cardi)	= 95 - 50	= 45
V(Alicia, Bob, Drake) - V(Bob, Drake)	= 80 - 40	= 40
V(Alicia, Cardi, Drake) - V(Cardi, Drake)	= 70 - 25	= 45
V(Alicia, Bob, Cardi, Drake) - V(Bob, Cardi, Drake)	= 100 - 60	= 40

$$\text{Alicia} = 40/4 + (45 + 35 + 40)/12 + (45 + 40 + 45)/12 + 40/4 = 40.8333$$

Doing the same calculations for the other players gives their Shapley values, or the proportion of the $100 total each should receive:

Alicia	Bob	Cardi	Drake
40.83	31.67	19.17	8.333

1.2   Donor Acceptance

We built two predictive models (RF, LogReg) for estimating the probability that a pediatric donor heart is accepted by a candidate. Interest is in understanding the features clinicians use to base their Accept/Refuse decisions. There are $n = 30,156$ offers and $p = 44$ predictor variables (features).

The most important predictors[1] are:
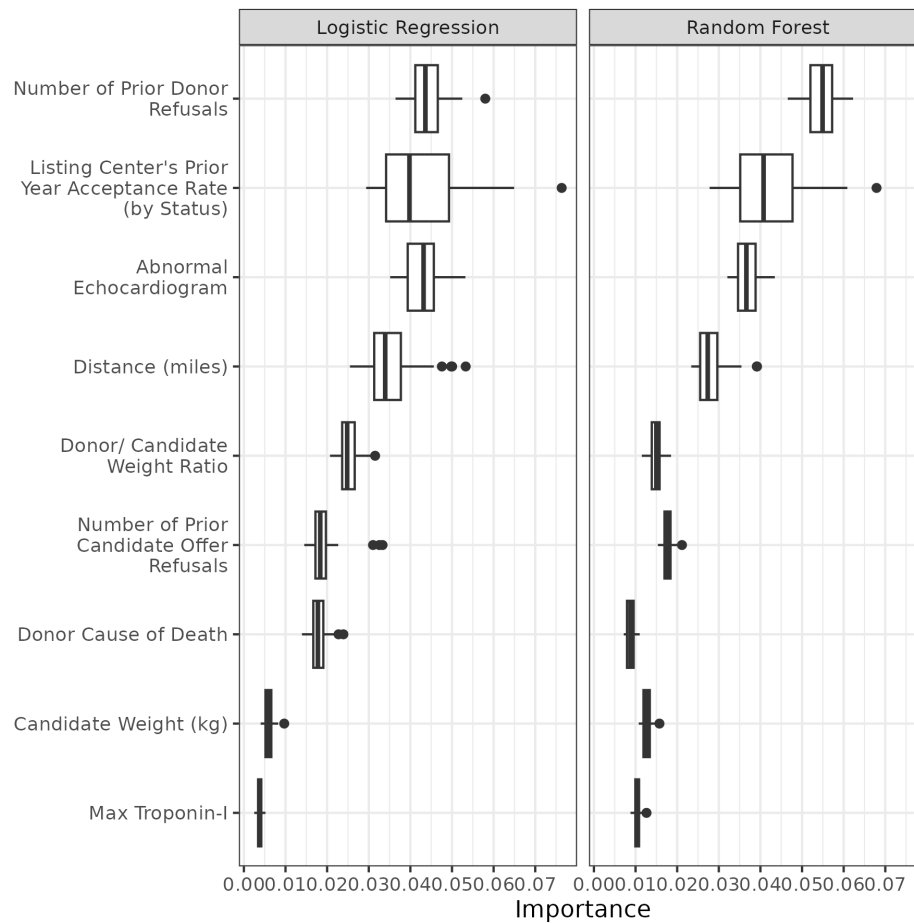
As a running example, let's consider a particular offer:

feature	value
NUM_REJECT_DON	1.00
LISTING_CTR_ACCEPT_RATE_PREV	0.35
ABNL_ECHO_CUM	0.00
DISTANCE	441.74
WEIGHT_RATIO	1.06
prior_offers_cand	1.00
COD_DON	Head Trauma
WEIGHT_CAND_KG	38.40
TROPONINI_max	-1.00
...	...
f_hat	0.783

---

[1] according to SHAP importance

1.3 Shapley values for explaining a prediction

Training data D is used to build a predictive model $\hat{f}(x)$, where $x = [x_1, \ldots, x_p]$. Shapley (or SHAP) values in this context attempt to find functions $\phi_0, \phi_1, \ldots, \phi_p$ such that the prediction of x can be explained by the linear sum:

$$\hat{f}(x) = \phi_0(x) + \sum_{j=1}^{p} \phi_j(x)$$

The value $\phi_j(x)$ is the Shapley value for the jth predictor variable and $\phi_0(x) = \phi_0 = E_X[\hat{f}(X)]$.

The connection to the original Shapley coalitions is that each feature is a player, the total value is $\hat{f}(x)$. Let $S \subseteq \{1, 2, \ldots, p\}$ indicate the set of features in a coalition. The trick is determining how to form the value function $V_x(S)$, where the subscript x is a reminder that this is the value function for a prediction at x. This is the big idea:

$$V_x(S) = E_{\bar{S}}[f(x_S, X_{\bar{S}})] - E_X[\hat{f}(X)]$$

where $X_{\bar{S}}$ is from the distribution that is assumed independent from S. That is, we treat the values of the features in S as known, but pretend the other features values, those in the complement $\bar{S}$, are missing. The expectation is over the missing feature values. Note that $\phi_0 = E_X[\hat{f}(X)]$ is the average prediction (or the prediction when all features are missing).

Let's come back to our offer example. The average prediction is 0.116. Consider the coalition $S = \{\texttt{NUM\_REJECT\_DON}, \texttt{LISTING\_CTR\_ACCEPT\_RATE\_PREV}\}$.

feature	value
NUM_REJECT_DON	1.00
LISTING_CTR_ACCEPT_RATE_PREV	0.35
ABNL_ECHO_CUM	_____
DISTANCE	_____
WEIGHT_RATIO	_____
prior_offers_cand	_____
COD_DON	_____
WEIGHT_CAND_KG	_____
TROPONINI_max	_____
All other features	...
f_hat	_____

We need to fill in the missing feature values with their expected values. But we have some options: (i) condition on $X_S = x_S$, (ii) treat X_S and $S_{\bar{S}}$ as independent, (iii) treat all features as independent, etc.

Let's consider option (ii), treating X_S and $S_{\bar{S}}$ as independent. We can estimate the expected value by averaging predictions from replacing the missing values with values from the training data D.

$$E[\hat{f}(X) \mid X_S = x_S, X_{\bar{S}} = \text{missing}] \approx \frac{1}{n}\sum_{i=1}^{n} \hat{f}(Z_i^S) \qquad \text{where } Z_i^S = [x_S, X_{i,\bar{S}}]$$

where $X_{i,\bar{S}}$ are feature values for $\bar{S}$ from observation i. Another way to express this is $Z_i^S = xw_S + X_i(1 - w_S)$ where $w_S = [1, 1, 0, 0, \ldots, 0]$ indicates which features are in coalition S.

> **Note**
>
> You need the prediction function $\hat{f}(z)$ to be able to calculate the prediction for any z.

| feature	x_S	offer: 1	offer: 2	offer: 3	offer: 4
NUM_REJECT_DON	1.00	_____	_____	_____	_____
LISTING_CTR_ACCEPT_RATE_PREV	0.35	_____	_____	_____	_____
ABNL_ECHO_CUM	_____	0.50	1.00	0.00	0.00
DISTANCE	_____	536.99	457.03	210.61	433.86
WEIGHT_RATIO	_____	0.81	1.27	0.76	2.06
prior_offers_cand	_____	0.00	0.00	0.00	0.00
COD_DON	_____	CVA	Infection (CNS)	Head Trauma	Head Trauma
WEIGHT_CAND_KG	_____	3.10	6.37	9.00	3.30
TROPONINI_max	_____	5.00	0.10	2.78	2.78
All other features	...	...	...	...	...
f_hat	_____	0.097	0.006	0.188	0.149

If we repeat this procedure for all subsets $S \subseteq \{1, 2, \ldots, p\}$ and use the formula to calculate all the Shapley values for offer x, we get:

feature	value	Shapley
NUM_REJECT_DON	1.00	0.094
LISTING_CTR_ACCEPT_RATE_PREV	0.35	0.202
ABNL_ECHO_CUM	0.00	0.086
DISTANCE	441.74	0.033
WEIGHT_RATIO	1.06	0.022
prior_offers_cand	1.00	0.039
COD_DON	Head Trauma	0.022
WEIGHT_CAND_KG	38.40	0.007
TROPONINI_max	-1.00	0.020
All other features	...	0.143
f_hat	0.783	0.116

Here are the Shapley values for a different offer. This offer has an estimated acceptance probability of only 0.008, lower than average.

feature	value	Shapley
NUM_REJECT_DON	24.00	-0.046
LISTING_CTR_ACCEPT_RATE_PREV	0.05	-0.020
ABNL_ECHO_CUM	0.00	0.012
DISTANCE	634.54	-0.019
WEIGHT_RATIO	0.93	0.001
prior_offers_cand	0.00	0.010
COD_DON	Anoxia	-0.006
WEIGHT_CAND_KG	23.65	-0.008
TROPONINI_max	5.41	-0.009
All other features	...	-0.021
f_hat	0.008	0.116

Here are the Shapley values for a sample of 300 random offers:
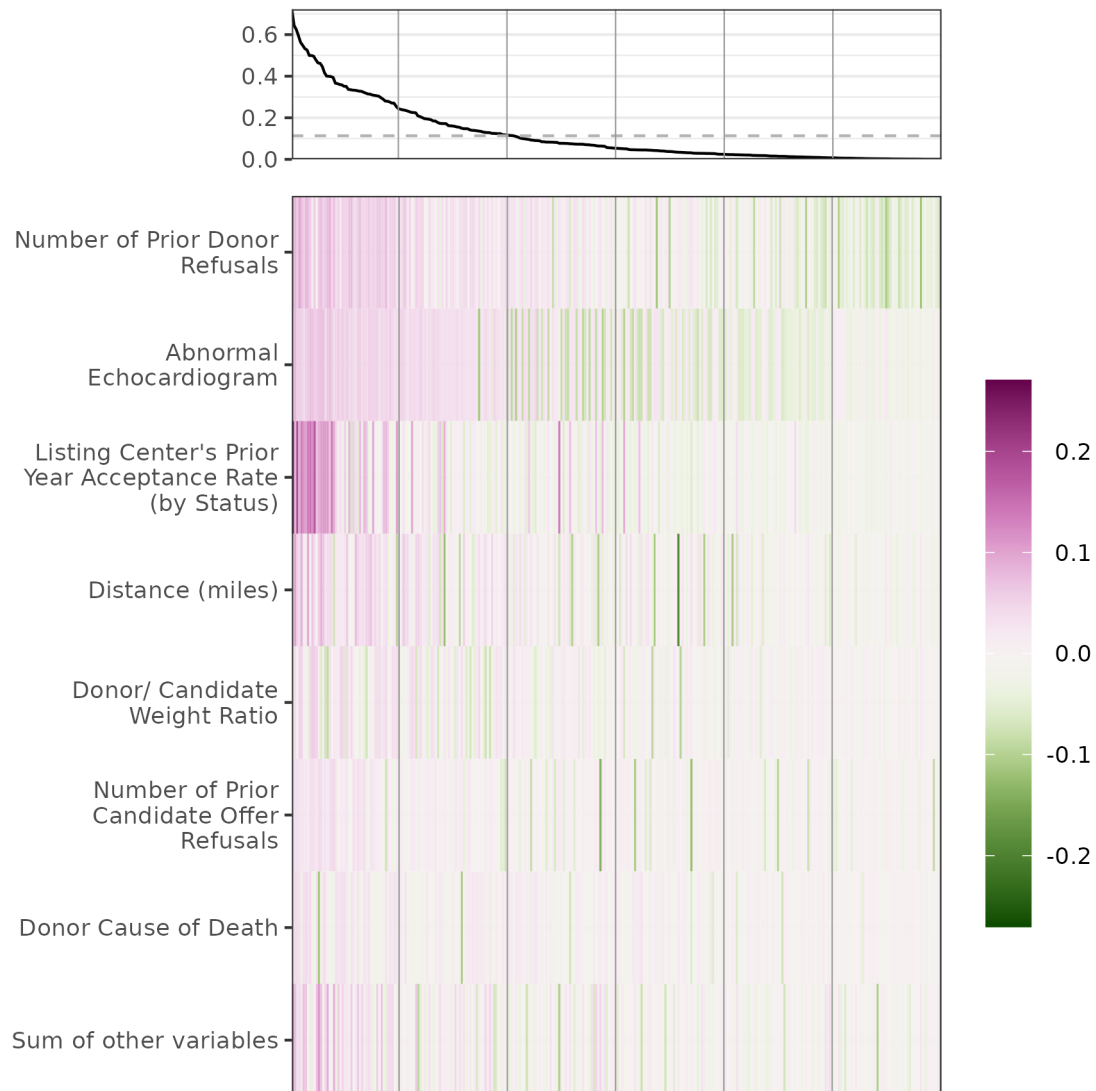
1.3.1   Estimating Shapley Values

There are too many features ($p = 44$) to exhaustively calculate all subsets (at least for my level of patience). Also, note that we need to make repeated calls to the prediction function, so the speed of prediction will play a role in estimating the Shapley values.

1. Shapley Sampling

Štrumbelj and Kononenko (2014) propose an approximation with Monte-Carlo sampling (see IML book 9.5.3.3):

$$\hat{\phi}_j(x) = \frac{1}{M} \sum_{k=1}^{M} \hat{f}(Z_m^{S_m \cup j}) - \hat{f}(Z_m^{S_m})$$

where S_m is a random subset. For each iteration, two samples are drawn (i) random data instance (i.e., a row of data) (ii) random set of features S_m that does not include j. Form $Z_m^{S_m \cup j}$ and make a prediction. Then replace feature j with x_j and make another prediction. The difference between the two predictions is the gain/importance of feature j to the prediction.

Here is an example using four samples:
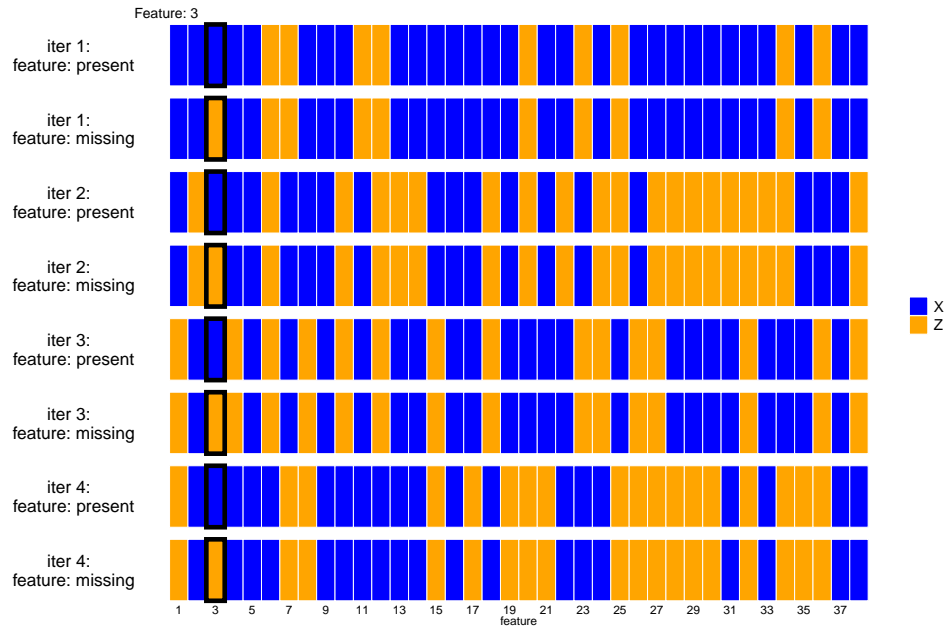
2. KernSHAP

Recognizing that the Shapley values can be specified in a linear model

$$\hat{f}(x) = \phi_0(x) + \sum_{j=1}^{p} \phi_j(x)$$

KernelSHAP is an approach to estimate the Shapley values using weighted linear regression:

$$\sum_{S} w(S) \left(\hat{f}(x_S) - \phi_0 - \sum_{j} \phi_j \right)^2$$

using the weight function

$$w(S) = \frac{1-p}{\binom{p}{k_S} k_S (p - k_S)}$$
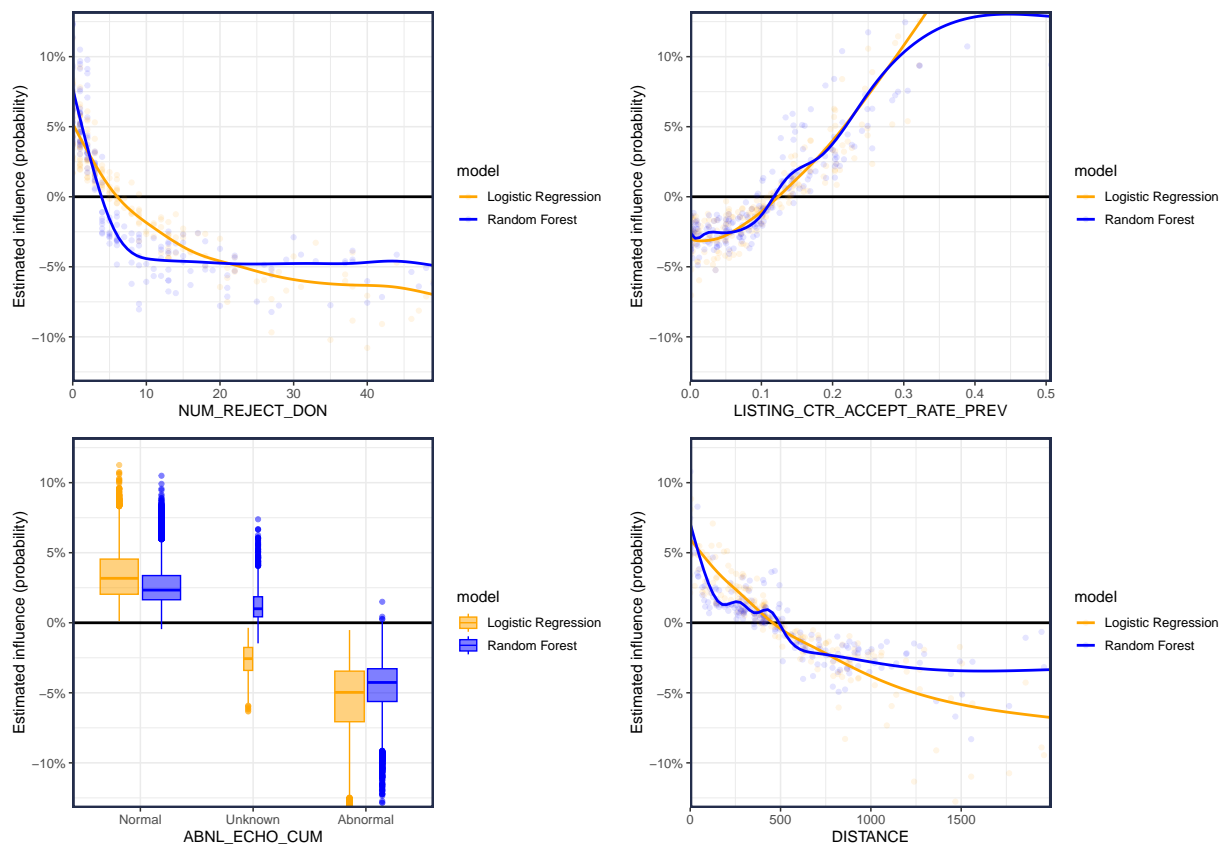
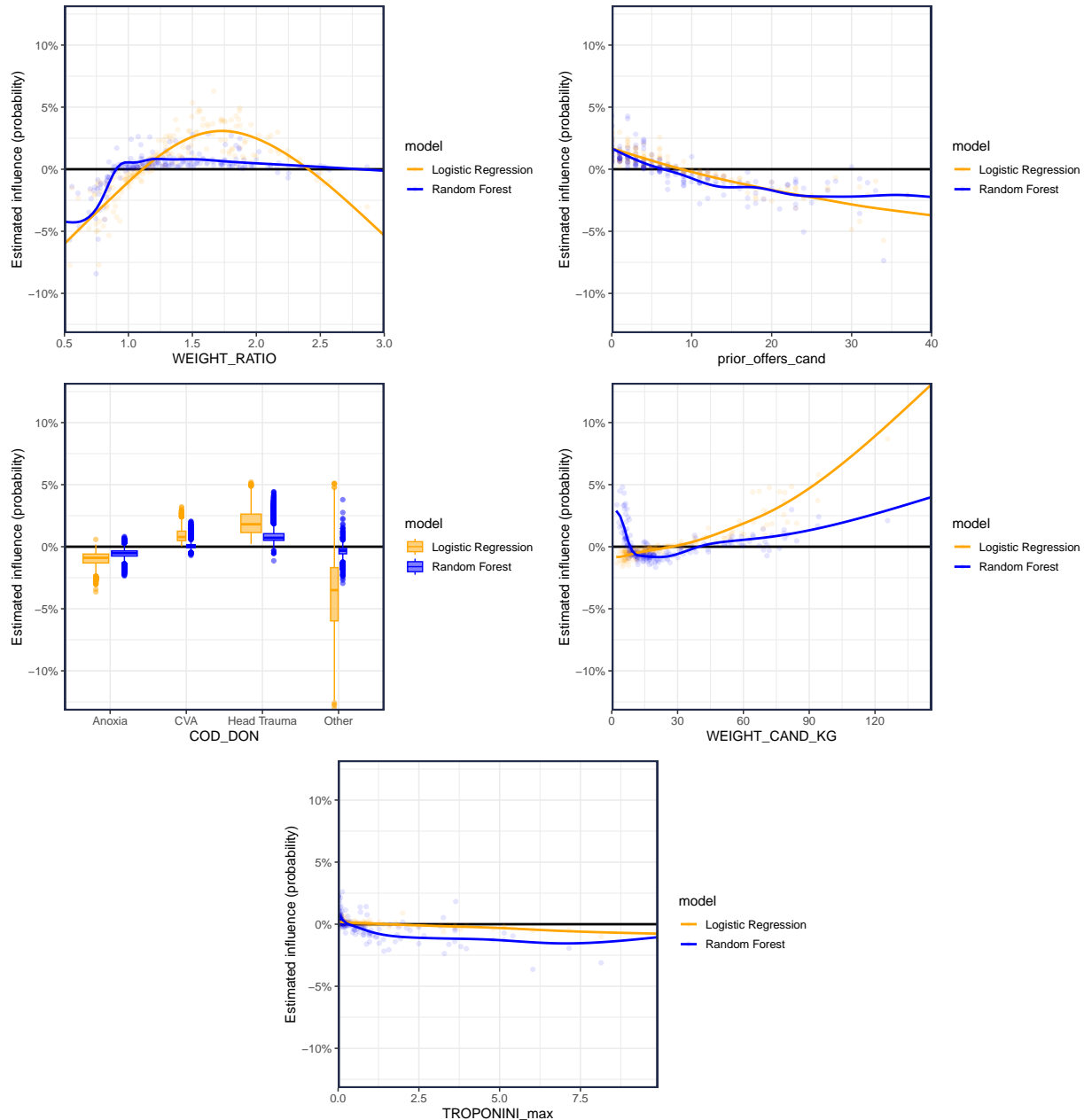3. Model specific methods

TreeSHAP, DeepSHAP, MaxSHAP

1.4 Shapley Effects Plots

We can estimate the Shapley values for a sample of the data and make marginal effects plots. The following scatterplots are formed from

$$\{x_{ij}, \phi_{ij}\}$$

where x_{ij} is from the data and ϕ_{ij} is the jth shapley value for observation i.

1.5 Shapley Feature Importance

The larger a feature's absolute value of the Shapley values (in the data) the more influence it has on the predictions. The SHAP feature importance score is given by the average absolute value (over the data).

$$I_j = \frac{1}{n} \sum_{i=1}^{n} |\phi_{ij}|$$

I think variance or standard deviation would also be a good metric for variable importance.

Looking back up to the plots. The feature importance is a measure of the average deviation from the points (shapley values) and the horizontal line at 0.

2   Permutation Feature Importance

Shapley feature importance measures how much the *predictions* change if the features are manipulated (i.e., treated as missing). An alternative is to measure how much the *performance* of the predictions change if the features are manipulated. Another way to say this, is to modify the feature(s) and measure the change in average loss. Using the notation from above, let Z_S be a new dataset where the features in S are randomly permuted/shuffled. This is equivalent to treating as missing as we did in Shapley. But there are two primary ways to measure the performance change: (1) shuffle the data and re-fit the model and (2) use the original model and only shuffle before prediction. These are detailed below:

2.1   Approach 1: re-fit model

Let $S = j$ be a single variable.

0. Create a hold-out set X_{test}. Fit a model to the training data $\hat{f}(X)$, predict on the hold-out data, and evaluate the predictions $L(Y_{\text{test}}, \hat{f}(X_{\text{test}}))$. If we manipulate any features, the performance should decrease.

1. Create $Z_j = [X_{1:(j-1)}, P(X_j), X_{(j+1):p}]$ where $P(X_j)$ is a randomly permuted vector of X_j.

2. Fit model $\hat{f}_j(x)$ using data Z_j

3. Prediction on a hold-out set X_{test} and evaluate the loss $L(Y_{\text{test}}, \hat{f}_j(X_{\text{test}}))$. The change in loss is the feature importance score. I.e., importance of feature $j = L(Y_{\text{test}}, \hat{f}_j(X_{\text{test}})) - L(Y_{\text{test}}, \hat{f}(X_{\text{test}}))$.

4. Repeat steps 1:3 (or 0:3) multiple times. Also, repeat 1:3 using different features.

This approach assesses how important a feature is for making a good prediction. However, there are some issues to be aware of. Suppose you have a set of highly correlated predictors. Their importance will be near zero because you can remove any of them, and one of their correlated partners will take up the slack. If the cost of collecting features is high, then this can be a good way to remove some and maintain high predictive performance.

Of course, you can always try jointly permuting subsets $|S| > 1$ (like for Shapley values) to get the joint importance.

2.2   Approach 2: shuffle before prediction

This approach is less computational. You only shuffle the hold-out data.

0. Create a hold-out set X_{test}. Fit a model to the training data $\hat{f}(X)$, predict on the hold-out data, and evaluate the predictions $L(Y_{\text{test}}, \hat{f}(X_{\text{test}}))$. If we manipulate any features, the performance should decrease.

1. Permute one (or more) columns of the *hold-out* data.

2. Predict and evaluate performance. The change in loss is the feature importance score.

3. Repeat multiple times and take average.

This approach assesses how important each feature is to *the model learned from the training data*. So in that set of correlated predictors, maybe only one or two will get used. Those features will appear important, but the others won't.

2.3   Boruta feature selection

Boruta is a *False Selection Rate (FSR)* feature selection method originally designed for random forest (or any tree-based models).

> **Boruta for trees**
>
> - Introduce additional shuffled features (shadow features). This increased the number of predictor variables from p to $2p$.
> - Calculate importance scores for all features (using the built-in split-based importance metrics)
> - Record the "hits": all original features with importance scores greater than *max importance from all shuffled features* (these features are deemed important).
> - Repeat the process M time (100 by default; or sequential).
> - Determine which predictors have significantly more "hits" than expected under null of not-important.