An Overview of SHAP-based Feature Importance Measures and Their Applications To Classification

Kyle Vedder
kvedder@seas.upenn.edu

May 8, 2020

1 Regression

1.1 Regression Feature Importance Problem

The *Regression Feature Importance Problem* (RFIP) states:

Definition 1 (Regression Feature Importance Problem)**.**

Given 1. N observations contained in P feature vectors $X_1, X_2, \ldots, X_P$, each of dimension N, and a model f which takes P features defined in (*Feature Index*, *Observation Index*) format. The value of feature p of observation n is obtained via $v((p,n))$. The output of f for the nth observation is $f(\{(1,n),(2,n),\ldots,(P,n)\}) \in \mathbb{R}$. f has an expected value over the N observations $\mu := \frac{1}{N}\sum_{n=1}^{N} f(\{(1,n),(2,n),\ldots,(P,n)\})$.

Output 1. Φ, an $N \times P$ matrix of importance values, where the nth row contains the importance value for the nth observation, with columns 1 through P corresponding to features 1 through P.

1.2 Regression SHAP Values

Regression SHAP (SHapley Additive exPlanations) is a *class* of additive feature importance measures to explain individual observations for regression. Formally:

Definition 2 (Regression SHAP)**.**

Given 2. feature observations $(1,n),(2,n),\ldots(P,n)$.

Output 2. Regression SHAP values $\phi_{1,n}, \phi_{2,n}, \ldots, \phi_{P,n}$.

Thus, Regression SHAP values can be used to solve the RFIP by computing Regression SHAP values for each observation, with the Regression SHAP value for each feature corresponding to the appropriate column in Φ. The general form for computing a Regression SHAP value for observation n, feature i is

$$\phi_{i,n} = \sum_{S \subseteq (F \setminus (i,n))} \frac{|S|!(|F| - |S| - 1)!}{|F|!} \left[f'(S \cup \{(i,n)\}) - f'(S) \right] \tag{1}$$

where $F = \{(1,n),(2,n),\ldots(P,n)\}$ and f' is a proxy function for the model f. It's important to note two things about Equation 1:

1. Not all feature values in the invocation of f' are specified, meaning the missing features must be set to some value before f' can invoke the model f.

2. $\sum_{S \subseteq (F \setminus (i,n))}$ iterates over the powerset of features ($2^{|F|-1}$ sets), making the runtime of the equation as written exponential in the number of features.

The fact that these challenges can be handled differently is what makes Regression SHAP a *class* of solutions. We present two approaches to solving Challenge 1, Regression IntegSHAP (Section 1.3) and Regression CondSHAP (Section 1.4), and these approaches along with additional assumptions lend themselves to solving Challenge 2 (Section 1.5).

1.3 Computing Regression IntegSHAP Values

As discussed in Section 1.2, the set S in the invocation of $f'(S)$ may not fully specify the values of all features (Equation 1), thus requiring translation to invoke the model f. We present Regression IntegSHAP, which operates by computing Regression SHAP values via integration over the features not fixed by the model. This integration, stated formally for observation n:

$$f'(S) := \frac{1}{N} \sum_{k=1}^{N} (f(S \cup \{((i,k) \mid \forall i : 1 \leq i \leq P \wedge (i,n) \notin S\}) \tag{2}$$

This approach is isomorphic to the marginalization approach proposed to by Danzing et. al. [1], and thus can be viewed in the context of Pearl's *do* notation as *do*ing the fixed features, therefore breaking the correlations structure with the unfixed features and thus not influencing their distribution. This approach requires a linear sweep of the N observations, making the runtime to compute a Regression IntegSHAP value for a single feature of a single observation $O(2^{P-1}N)$, but it upholds several desirable axioms:

Axiom 1 (Efficiency)**.** *The sum of the Regression SHAP values across all features for observation n must equal the model's deviance from the model's mean for observation n, i.e. $\phi_{1,n} + \phi_{2,n} + \ldots + \phi_{P,n} = f(\{(1,n),(2,n),\ldots(P,n)\}) - \mu$.*

Axiom 2 (Dummy)**.** *A feature that does not change the model output has a Regression SHAP value of zero.*

Notably, Regression IntegSHAP does *not* uphold the Shapley Symmetry axiom (if (i,n) and (j,n) have identical impact on the model output, $\phi_{i,n} = \phi_{j,n}$). Using the example of failure of symmetry for another approach in Sundararajan et. al. [2]: consider two Bernoulli random variables $X1$ and $X2$, where $X1$ is 1 with probability p (otherwise 0) and $X2$ is 1 with probability q (otherwise 0); naturally, $\mathbb{E}[X1] = p$ and $\mathbb{E}[X2] = q$. Additionally, let model $f(X1, X2) = X1 + X2$. Under Shapley's Symmetry axiom, given the observation $(X1 = 1, X2 = 1)$ both $X1$ and $X2$ should derive the same feature importance as both contribute equally to the model; however, note that if $p \neq q$, they will not with Regression IntegSHAP:

$$\begin{aligned}
\phi_{X1} &= \frac{1}{2}(f(X1, X2) - f(\mathbb{E}[X1], X2)) + \frac{1}{2}(f(X1, \mathbb{E}[X2]) - f(\mathbb{E}[X1], \mathbb{E}[X2])) \qquad (3)\\
&= \frac{1}{2}(f(1,1) - f(p,1)) + \frac{1}{2}(f(1,q) - f(p,q))\\
&= \frac{1}{2}(2 - p - 1) + \frac{1}{2}((1+q) - p - q)\\
&= (1-p)\\
\phi_{X2} &= \frac{1}{2}(f(X1, X2) - f(X1, \mathbb{E}[X2])) + \frac{1}{2}(f(\mathbb{E}[X1], X2) - f(\mathbb{E}[X1], \mathbb{E}[X2])) \qquad (4)\\
&= \frac{1}{2}(f(1,1) - f(1,q)) + \frac{1}{2}(f(p,1) - f(p,q))\\
&= \frac{1}{2}(2 - 1 - q) + \frac{1}{2}(p + 1 - p - q)\\
&= (1-q)
\end{aligned}$$

Danzing et. al. argue this is sensible behavior [1]; if a feature is further from its expected value, then that feature is more unusual and thus deserves more importance. This rational appears sensible for features with unimodal, symmetric distributions of similar variance and outputs dependant on feature scale, but breaks down as feature distributions and models get more complex. Consider $X1$ and $X2$ to be zero mean Gaussian features with variance σ_1 and σ_2, respectively, and $\sigma_1 \gg \sigma_2$. Let the model be a regression tree splitting on their means and predicting $\text{sign}(X1) \cdot \text{sign}(X2)$ in its four leaves; this model is invariant to the scale of $X1$ and $X2$, yet if $X1$ and $X2$ both draw a sample one standard deviation from their mean, Regression IntegSHAP will much more heavily weight $X1$ than $X2$, as $X1$'s sample is significantly further from its mean, yet each feature's contribution is equally significant. The problem in this scenario would be alleviated by data whitening as a preprocessing step (making $\sigma_1 \approxeq \sigma_2$), but more elaborate scenarios cannot be fixed by whitening.

Additionally, Regression IntegSHAP does not maintain cotenability in the samples it feeds into the model; in fact, because it fixes all subsets of features and integrates the unfixed features, this approach is guaranteed to break all feature correlations at least once in its inputs to the model. This will require extrapolation by the model, possibly producing wild results in high capacity, high variance models.

Despite these two caveats, Regression IntegSHAP used in the context of the RFIP produces an output matrix Φ with several desirable properties. For instance, the sum of each row in the RFIP's output Φ is the difference of that row's prediction from μ (Axiom 1). The sum of the differences of all n observations from μ is zero (this follows from the definition of μ), thus the sum of the elements of Φ is zero. Φ also allows for the study of the impact feature i has on the outcomes of any subset of the N observations N' via $\sum_{n \in N'} |\phi_{i,n}|$; this value measures how much feature i changed the outcomes of the model relative to μ for the observations N'.

1.4 Computing Regression CondSHAP Values

Section 1.3 presents Regression IntegSHAP, a method for fully specifying feature values in translating f' to f (Equation 1); however, Regression IntegSHAP breaks feature correlations on its inputs to the model, potentially causing wild extrapolations from high variance models. Regression CondSHAP solves this problem by fixing values for a subset of features, and then restricting evaluation

to the subset of N observations with those fixed feature values. This approach, stated formally for observation n:

$$f'(S) := \frac{1}{N'} \sum_{o \in N'} f(o) \tag{5}$$

where $N' := \{\{(1,i),(2,i),\ldots,(P,i)\} \mid \forall i \in \{1,2,\ldots,N\}, \forall (p,n') \in S : v((p,n)) = v((p,i))\}$, i.e. N' is the set of observations which have feature values which match the fixed feature values.

This approach was first introduced in 2017 by Lundberg et. al. [3] and the values it computes originally bore the title of simply *SHAP values*. The values computed by Regression CondSHAP fundamentally differ from those computed by Regression IntegSHAP; by conditioning on certain feature values, the distribution of correlated unconditioned features changes. Under the strong assumption that all features are perfectly uncorrelated, this problem is theoretically alleviated and Regression CondSHAP collapses to Regression IntegSHAP, but in practice even if this assumption is true, for finite observations there is still the potential that the empirical distributions for the features of all observations and the features of observations matching the condition still slightly differ. The degenerate case for this occurs when the fixed features are unique to a single observation, causing Regression CondSHAP to collapse the unfixed feature distribution to the single values in that observation. This can be triggered in otherwise well behaved datasets by adding a small amount of noise to the features of each observation, causing uniqueness in feature values and resulting in this collapse for every observation. As a result of these differences, while Regression CondSHAP maintains Regression IntegSHAP's Efficiency axiom (Axiom 1), the Dummy axiom (Axiom 2) is not preserved [2]. Additionally, like Regression IntegSHAP, Regression CondSHAP does not uphold Shapley's Symmetry axiom; it will produce identical results in the example in Section 1.3.

1.5 The many values of Regression TreeSHAP

Regression TreeSHAP is a Regression SHAP approach that assumes tree model form in order to speed up the evaluation of Equation 1, allowing Regression TreeSHAP to run polynomial in the size of the trees. Both the 2018 ArXiv paper [4] and the the 2020 Nature paper [5] present the same algorithm for Regression TreeSHAP (Algorithms 1 and 2 in both papers) which clearly demonstrate that Regression TreeSHAP computes Regression CondSHAP values by computing the unconditioned feature distributions from leaves accessible from the conditioned values; thus, we dub this method Regression CondTreeSHAP. However, following the publication of Danzing et. al. [1], the SHAP software package[1] was updated with a mode that computes Regression IntegSHAP values, using the training data ("background dataset") to represent the empirical distribution of features rather than the distribution found in the model's leaves[2]. While this change now allows Regression TreeSHAP to uphold the Dummy axiom (Axiom 2), it increases the computational complexity of Regression TreeSHAP to depend linearly on the number of points in the background dataset. We dub this method Regression IntegTreeSHAP.

[1] `https://github.com/slundberg/shap`

[2] This is now the default behavior when `shap.TreeExplainer` is provided a background dataset.

2 Classification

2.1 Binary Classification Feature Importance Problem and Binary SHAP

The Binary Classification Feature Importance Problem is a special case of k-class Classification Feature Importance Problem with a class $c \in \{0, 1\}$. As a result, we can use the definition of Regression Feature Importance Problem (Definition 1) to form the Binary Classification Feature Importance Problem (BCFIP) with the single modification of restricting the range of f to $[0, 1]$, corresponding to $p_{c=0}(n)$, i.e. the probability of class 0, for observation n. As a result of this change, μ naturally becomes the fraction of the N observations with $c = 0$. Thus, we can use Equation 1 to construct Binary SHAP, and then use Equation 2 identically to Section 1.3 to construct Binary IntegSHAP and the results of Section 1.5 to construct Binary TreeSHAP. As there are no required changes to Regression IntegSHAP to construct Binary IntegSHAP or Binary IntegTreeSHAP, both continue to uphold Efficiency (Axiom 1) and Dummy (Axiom 2).

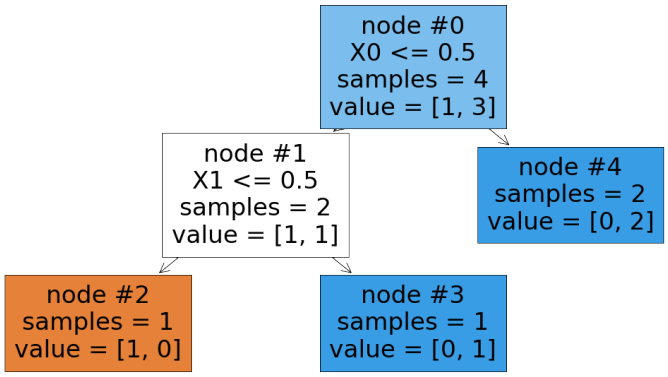
Consequently, the Binary IntegSHAP values for observation n represent how much each feature changes $p_{c=0}(n)$, with $\phi_{1,n} + \phi_{2,n} + \ldots + \phi_{P,n} = p_{c=0}(n) - \mu$, i.e. the change of observation n's probability of $c = 0$ from the baseline. Thus, it follows that each ϕ is limited in the range it can vary based on μ as each ϕ represents a change in probability for the given feature and thus cannot go above 1 or below 0 ($\forall \phi : -\mu \leq \phi \leq 1 - \mu$). By summing all $|\phi|$ for a particular feature p ($\frac{1}{N} \sum_{n \in \{1,2,\ldots,N\}} |\phi_{p,n}|$) we can measure how much variance feature p causes in the final output. This computation is more informative than a similar computation on a RFIP Φ because the magnitude of a Regression ϕ is potentially unbounded for an outlier, dramatically inflating this sum, whereas for a BCFIP Φ the magnitude of a Classification ϕ is bounded, bounding the the inflation any outlier can force on this sum. Additionally, you can drill down and understand how much feature p just causes the result to go up or just go down for a subset of observations $N' \subseteq \{1, 2, \ldots, N\}$ by computing $\frac{1}{|N'|} \sum_{n \in N'} \max(\phi_{p,n}, 0)$ and $\frac{1}{N} \sum_{n \in N'} \min(\phi_{p,n}, 0)$, respectively, and with the difference between the two measuring the skew of that feature; this is useful in scenarios where you want to further understand false positives or false negatives for a subset of observations in isolation. Furthermore, like with RFIP's Φ computed via Regression IntegSHAP, the sum of the elements of BCFIP's $\Phi_{c=0}$ computed via Binary IntegSHAP is 0 and the absolute values of Binary SHAP values for a given feature and any subsets of observations can be summed to provide a general notion of influence. Finally, $\Phi_{c=0} = -\Phi_{c \neq 0}$, as the elements of Φ measure the change in probability of $c = 0$ from the mean, so naturally when any value in $\Phi_{c=0}$ goes up, the associated value in $\Phi_{c \neq 0}$ goes down by the same amount, and vice versa.

2.2 k-class Classification Feature Importance Problem and k-class SHAP

The k-class Classification Feature Importance Problem has classes $\{1, 2, \ldots, k\}$, but can be treated as k one-vs-rest BCFIPs. As a result, kCFIP produces k BCFIP matrices $\Phi_{c=1}, \Phi_{c=2}, \ldots, \Phi_{c=k}$, each modeling the probability of its associated class k. Interestingly, $\forall I \subset \{1, 2, \ldots, k\}, J = \{1, 2, \ldots, k\} \setminus I : \Phi_{c=I} = -\Phi_{c \neq I} = -\Phi_{c=J}$; this implies that the variance from μ for any subset of classes is perfectly mirrored by the remaining classes. This mathematically guarantees that any class merging or splitting will not change the importance measures of any class not materially involved in the split or merge.

3 Studying Binary and k-class Classification Trees

Section 1 and Section 2 provide the problem definitions and mathematical insights to allow for the exploration of Binary and k-class Classification Trees[3]. We begin with Figure 1, a simple binary classification tree where each feature is only referenced once in the model. Observations 0 and 1 are of the same class and follow the same path in the tree, thus receiving identical SHAP values; however, Observations 2 and 3 follow the same branch for Node 0, splitting on `X0` and diverge in Node 1, splitting on `X1`, but each assign different feature importances to the the splits on `X0` and `X1`. This result defies an intuitive expectation that observations which follow the same branch of the tree until the last split will assign the same feature importance to the earlier features, but this expectation is defied because Binary IntegSHAP fails to uphold Shapley's Symmetry axiom. More concretely, this result mathematically occurs because toggling just `X0` for Observation 2 does not change the outcome (Equation 6) but toggling just `X0` for Observation 3 drops the probability of $c = 0$ from 100% to 50% (Equation 7).



(a) Fit Binary Classification Tree ($c = 0$ is `[1, 0]`)

```
X0 = [ 1, 1, 0, 0]
X1 = [ 0, 0, 1, 0]
ys = [ 1, 1, 1, 0]
```

(b) Raw Data

	X0	X1
0	-0.375	0.125
1	-0.375	0.125
2	0.250	-0.500
3	0.500	0.250

(c) $\Phi_{c=0}$

Figure 1: Simple BCFIP for Class 0 (`[1, 0]`) computed with Binary IntegTreeSHAP

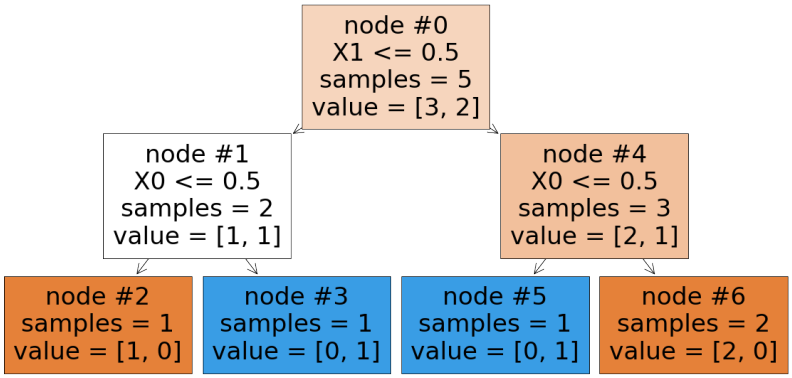
[3]All results from this section were generated via the Colab Notebook at `https://colab.research.google.com/drive/13PdABVy_-2Tufa7BWT0ikfD5a302xoPw`

$$
\begin{aligned}
I &= (1, 1, 0, 0) \\
J &= (0, 0, 1, 0) \\
\phi_{0,2} &= \frac{1}{2}\left[f(0,1) - \frac{1}{|I|}\sum_{i\in I} f(i,1)\right] + \frac{1}{2}\left[\frac{1}{|I|}\sum_{j\in J} f(0,j) - \frac{1}{|I|}\sum_{i,j\in(I,J)} f(i,j)\right] \qquad (6) \\
&= \frac{1}{2}\left[0 - 0\right] + \frac{1}{2}\left[\frac{3}{4} - \frac{1}{4}\right] \\
&= \frac{1}{4} \\
\phi_{0,3} &= \frac{1}{2}\left[f(0,0) - \frac{1}{|I|}\sum_{i\in I} f(i,0)\right] + \frac{1}{2}\left[\frac{1}{|I|}\sum_{j\in J} f(0,j) - \frac{1}{|I|}\sum_{i,j\in(I,J)} f(i,j)\right] \qquad (7) \\
&= \frac{1}{2}\left[1 - \frac{1}{2}\right] + \frac{1}{2}\left[\frac{3}{4} - \frac{1}{4}\right] \\
&= \frac{1}{2}
\end{aligned}
$$

Similar behavior also manifests itself in Figure 2, where two observations (Observation 0 and 3) are of the same class and take opposite paths on splits of the same value for both features; Observation 0 appears to indicate that `X1` is twice as important as `X0`, while Observation 3 appears to indicate that `X0` is twice as important as `X1`; however, the root cause of these differences is the result of relative sizing of the subsets of features; if one of Node 6's observations is removed, e.g. Observation 1, then the magnitude of all SHAP values becomes $\frac{1}{4}$.

From a usability perspective, Figure 1 and Figure 2 are both simple examples with a small number of binary features fed into simple models that lack significant feature interactions, and yet inspecting the Binary IntegSHAP values for any single Observation n fails to reveal any nuanced information without heavy visual inspection of the models themselves (Figure 1a and Figure 2a). However, by treating the $\Phi_{c=0}$ matrices as broadly informative for the model as a whole, we are able to gain interesting insights; for example, we can leverage the feature magnitude summing technique discussed in Section 2.1. In Figure 1 the $\Phi_{c=0}$ (Figure 1c), `X0` is assigned a larger value than `X1`, and by inspection of the model (Figure 1a) we see this makes sense as `X0` alone has more discriminatory power than `X1`. By comparison, in Figure 2 the $\Phi_{c=0}$ (Figure 2c), `X0` is assigned the same value as `X1`, and by inspection of the model (Figure 2a) we see this makes sense as `X0` alone has approximately the same discriminatory power as `X1`.

Figure 3 is a more complex example using SKLearn's Breast Cancer dataset which demonstrates the usefulness of the feature magnitude summing technique discussed in Section 2.1. The model (Figure 3a) is too large to understand without significant visual inspection, but the sum of the $|\phi|$ values for each feature (Figure 3b) allows us to quickly gain several insights into the model. First, only 13 of the initial 43 features impact the outcome of the model (Dummy, Axiom 2, guarantees that any ϕ with a non-zero value has some impact on the model outcome), and we can see that `worst radius` is the feature that most heavily sways model predictions, as it alone contributes more than a 20% average change in class relative to the base rate, with `worst concave points` and `worst concavity` following up at 11% and 9% respectively.



(a) Fit Binary Classification Tree ($c = 0$ is `[1, 0]`)

```
X0 = [ 1, 1, 1, 0, 0]
X1 = [ 0, 1, 1, 1, 0]
ys = [ 1, 0, 0, 1, 0]
```

(b) Raw Data

	X0	X1
0	-0.2	-0.4
1	0.2	0.2
2	0.2	0.2
3	-0.4	-0.2
4	0.2	0.2

(c) $\Phi_{c=0}$

Figure 2: Simple BCFIP for Class 0 (`[1, 0]`) computed with Binary IntegTreeSHAP

These same insights can then be readily scaled to mutli-class classification. Figure 4 shows a multi-class classification example using SKLearn's Wine data set. By stacking the classification sums of different features together but depicted with different colors as in Figure 4b, we are able to get a sense of how each feature impacts the prediction of each class. For example, `proline` is an impactful feature for $c = \{0, 1\}$, but not for $c = 2$, but we can still see that overall `proline` has the second highest impact on the accuracy of the model.

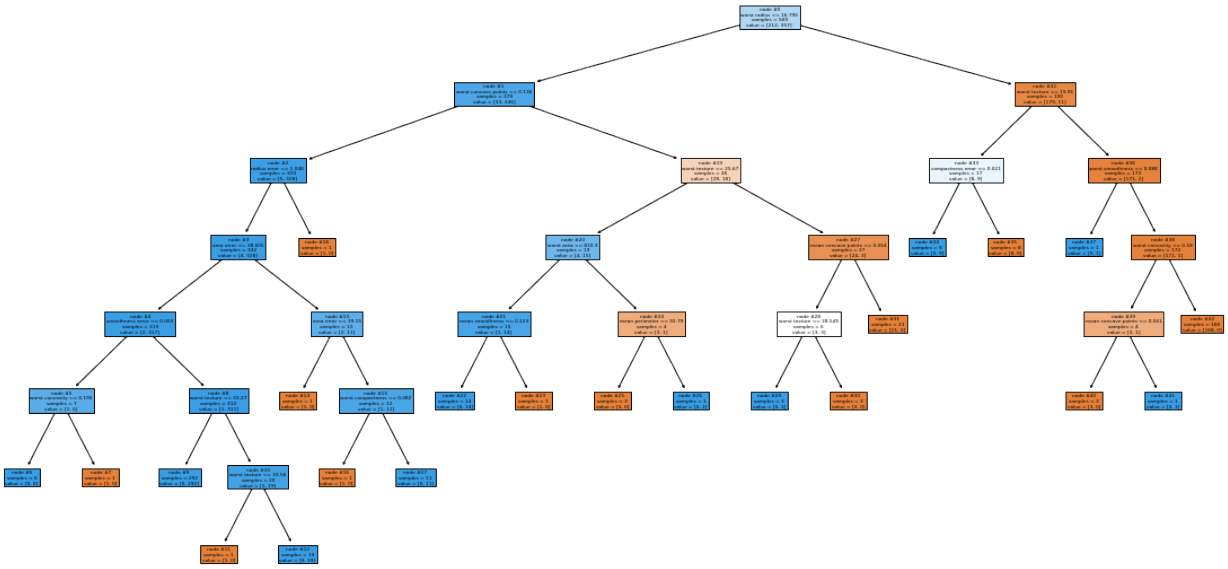
4 Bibliography

References

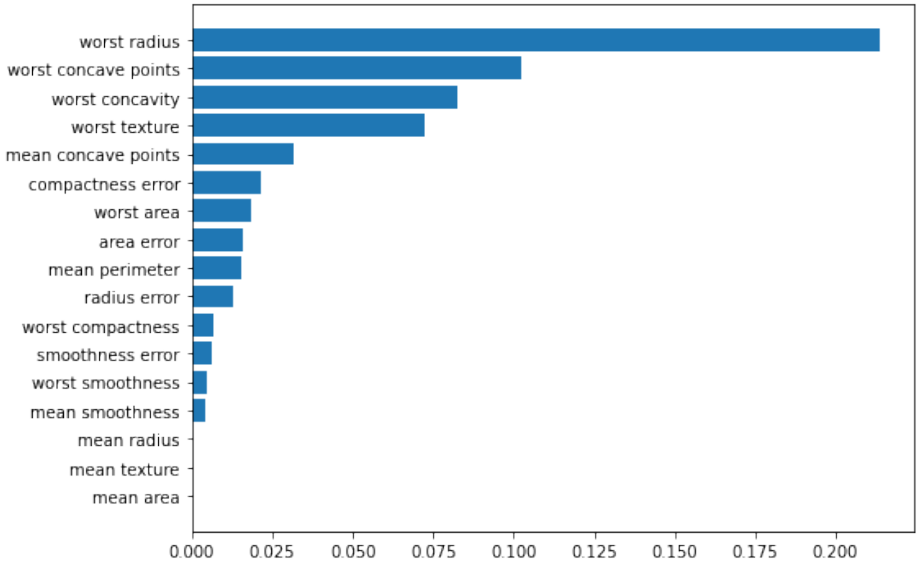
[1] D. Janzing, L. Minorics, and P. Blöbaum, "Feature relevance quantification in explainable AI: A causal problem," *ArXiv*, vol. abs/1910.13413, 2019.

[2] M. Sundararajan and A. Najmi, "The many Shapley values for model explanation," *ArXiv*, vol. abs/1908.08474, 2020.

[3] S. M. Lundberg and S.-I. Lee, "A Unified Approach to Interpreting Model Predictions," in *Advances in Neural Information Processing Systems 30* (I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, eds.), pp. 4765–4774, Curran Associates, Inc., 2017.

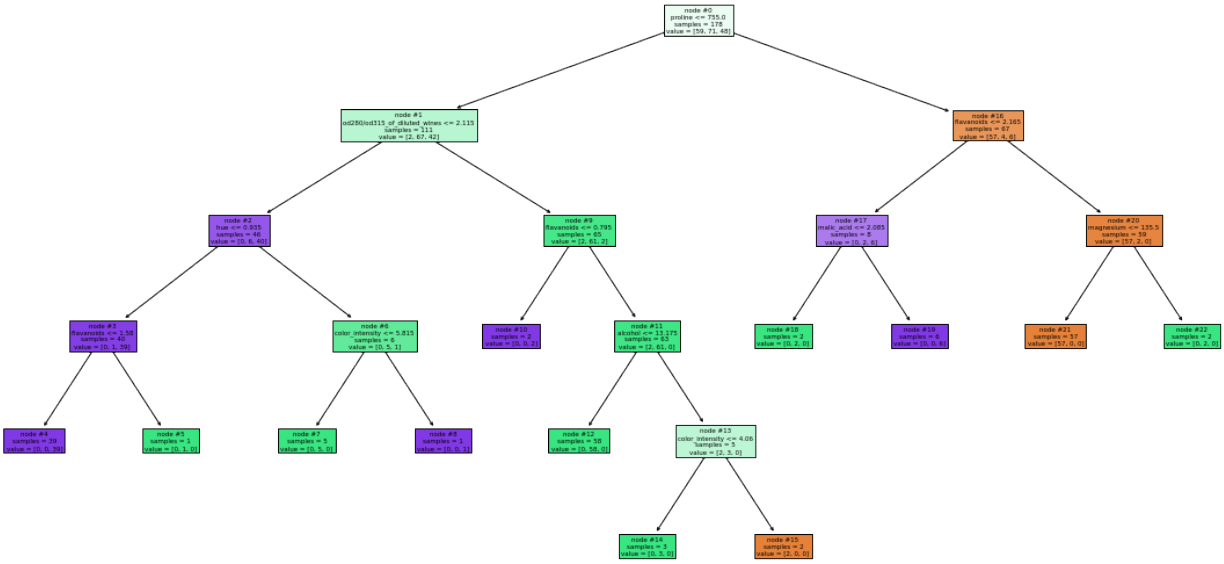


(a) Binary Classification Tree

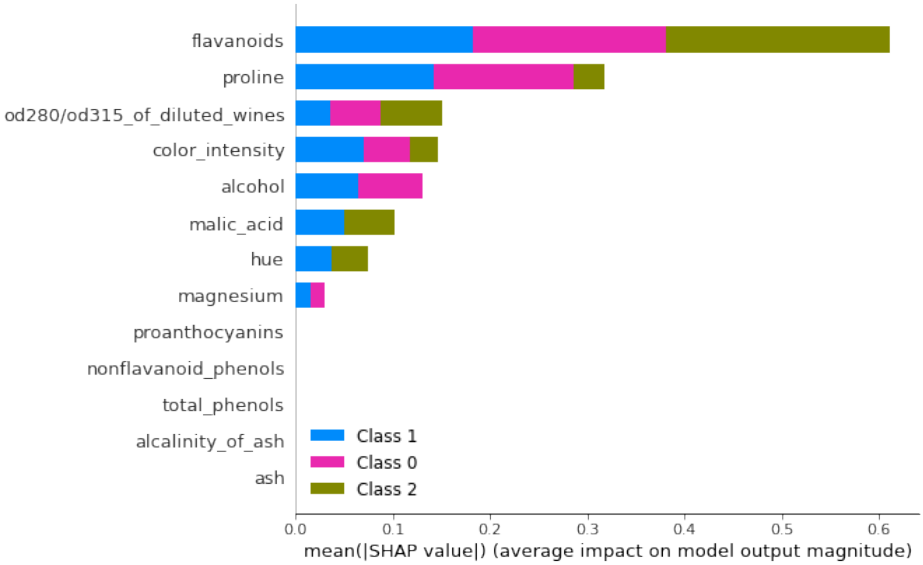


(b) Sum of $|\phi|$ values for each feature computed via Binary IntegTreeSHAP

Figure 3: Results from training a Binary Classification Tree on SKLearn's Breast Cancer dataset.



(a) Binary Classification Tree



(b) Sum of $|\phi|$ values for each feature computed via Binary IntegTreeSHAP

Figure 4: Results from training a Binary Classification Tree on SKLearn's Breast Cancer dataset.

[4] S. M. Lundberg, G. G. Erion, and S.-I. Lee, "Consistent Individualized Feature Attribution for Tree Ensembles," *ArXiv*, vol. abs/1802.03888, 2019.

[5] S. M. Lundberg, G. Erion, H. Chen, A. DeGrave, J. M. Prutkin, B. Nair, R. Katz, J. Himmelfarb, N. Bansal, and S.-I. Lee, "From local explanations to global understanding with explainable AI for trees," *Nature Machine Intelligence*, pp. 56–67, 2020.