1 Title Slide

Hello to everyone, my name is Spencer Matthews. In December of 2020 I graduated from Brigham Young University with a Bachelor's degree in Statistics, and I begin a PhD in Statistics at UC Irvine in September of this year. Today I will be presenting original research undertaken by myself and Brian Hartman while I was an undergraduate student at Brigham Young University. This research resulted in the mSHAP (or multiplicative SHAP) algorithm, which allows for computation of SHAP values on two-part models.

2 Motivation Slide

Since insurance is required for things such as owning a car or obtaining a mortgage it is important that pricing is neither discriminatory nor unfair. Two-part models are frequently used by actuaries to set insurance rates, and in order to achieve fairness these models must be explainable.

Newer "black-box" methods provide greater accuracy to pricing models, but make it difficult for actuaries to explain why the model made certain predictions. Recent research in the area of explainable machine learning has given powerful tools, such as SHAP values, to explain these "black-box" models. However, there is not a good methodology to explain the predictions of two-part models. We believe that using the SHAP values of the individual models, we can quickly approximate the SHAP values of the overall two-part model.

3 SHAP Value Introduction Slide

But first, a brief introduction to SHAP values. Lloyd Shapley pioneered what he called the Shapley value in the 1950s while doing research in cooperative game theory. His idea was to compute the contribution of each player to the overall outcome of the game.

In 2017, Scott Lundberg and Su-in Lee adapted Shapley values to the needs of machine learning with the idea of a SHAP value as computed by kernelSHAP. This powerful algorithm can estimate SHAP values from the predictions and the predictors for any set of data. It allows a black box model to become a glass box model where anyone can see how the model arrived at its prediction.

Currently, kernelSHAP is the most prominent possibility for explaining two-part models, but its computational cost grows exponentially as variables and observations are added. The high dimensional data available across millions of insurance policies makes using kernelSHAP a non-option for many actuaries.

TreeSHAP was proposed by Lundberg et al. in 2020 as a way to quickly compute exact SHAP values for tree-based algorithms. This set the stage for explaining two-part models consisting of tree-based algorithms, which is what the mSHAP algorithm does.

4 Definitions Slide

Before discussing the math behind mSHAP, we will define various terms as follows:

- First, we will define three models, f, g, and h and we will define h to be the product of f and g.
- Furthermore, we have an input matrix A which consists of n observations and p predictors. A_i represents the ith observation in A.
- For simplicity of notation, we define $\hat{x_i}, \hat{y_i}$ and $\hat{z_i}$ as f, g, and h of A_i, respectively. To denote the contribution of the jth covariate to the prediction, we will write s_{x_ij}.
- SHAP values depend on a baseline term, which is the average model prediction over the data. We will define this baseline term as μ_f, μ_g and μ_h for models f, g, and h.

This brings us to the property of local accuracy, which says that the ith prediction from model f will equal the sum of the SHAP values and the baseline term. In more general terms it can be written as this.

5 Local Accuracy Slide

Which in words means that the sum of the SHAP values and the expected model output must equal the model prediction. Local accuracy is an essential property of SHAP values as defined by Lundberg and Lee, but causes issues when we try to apply SHAP values to two-part models.

5.1 *SLIDE*

Consider a brief example.

5.2 *SLIDE*

In a very simple case we have two models with two covariates in our data set. The box here shows the breakdowns of the resulting predictions for the ith row of the data into their parts, based on the property of local accuracy. The prediction equals the sum of the mean model output and the variable contributions, denoted by s.

5.3 *SLIDE*

To compute $\hat{z}_i$, we multiply $\hat{x}_i$ and $\hat{y}_i$. However, in spite of what most of us wanted to believe at some point in our early education, the product of $\hat{x}_i$ and $\hat{y}_i$ is not as simple as it appears here.

In fact, jumping to a more general case with p predictors,

6 Expansion of Terms Slide

we see that the resulting expansion can be quite complex. Along the top margin we have the prediction $\hat{x}_i$ from model f, and along the left margin, we have the prediction $\hat{y}_i$ from model g. These two predictions have been expanded out into the contributions from each variable and the baseline term, per the local accuracy property. Each s in the margin is a SHAP value that contributes to a model prediction. Those in the top make up a single prediction from model f and those on the left do the same for model g. For our generalized two-part model, the prediction is the product of $\hat{x}_i$ and $\hat{y}_i$, the predictions from model f and g respectively. By definition, this is the sum of the table excluding the margins. When trying to compute the final SHAP values of the two-part model we see that the SHAP value of a single variable has been blended with those of all other variables.

6.1 *SLIDE*

Consider variable 1, whose contributions to the two model predictions are denoted in the margins by $s_{x_i 1}$ and $s_{y_i 1}$. It is apparent that there is an inherent difficulty in attributing a single contribution to the final prediction for this (or any) variable.

This brings us to our proposed approach, which involves first computing a modified contribution for each variable. Taking variable 1 as our example, this modified contribution is computed as the sum of the first row and the first column, where every term is divided by 2 except the terms containing a μ.

6.2 ***LOTS OF SLIDES***

This can be repeated for all predictors, covering every term in the table except $\mu_f\mu_g$.

Recall that for local accuracy to hold, we need contributions from each variable and the baseline model output, μ_h. However, $\mu_f\mu_g$ is not equal to μ_h, so we break $\mu_f\mu_g$ into two separate parts: μ_h and α, where α is an adjustment term equal to $\mu_f\mu_g - \mu_h$.

7 Proposed Approach Slide

Once we have our modified contributions, we add those to the mean of model h and α. Again, we know that the sum of α and μ_h must be equal to $\mu_f\mu_g$, so α is equivalent to $\mu_f\mu_g - \mu_h$. In total, we have the prediction of the two-part model equal to the sum of the modified contributions plus α plus the baseline term of the two-part model, where the modified contributions are calculated as previously mentioned.

Once this approach was derived, it became our task to distribute α back into the modified contributions, so that we could end up with the model prediction being equal to the sum of the contributions and the baseline term, as local acuracy requires.

8 Distributing α Slide

For this purpose, we considered four different ways of distributing α. The first was uniformly distributed, consisting of dividing alpha by the number of predictors. Our three other methods were based on the value of the modified contribution: raw value, absolute value, and squared value respectively.

In order to test how these different methods fared, we performed an in-depth simulation study.

9 Simulation Study

This simulation study involved generating fake data and computing two known responses, modelling the data, computing both kernelSHAP values and mSHAP values with alpha distributed in the four different ways, and then comparing the results. We scored the mSHAP values in relation to the kernelSHAP values, even though kernelSHAP is an estimate. The score was out of three, and was based on how many of the contributions had the same sign, how many had the same importance rank based on absolute value, and how close the actual values were.

In the end we obtained

10 Simulation Results Slide

these results. This table shows the average results for over 3,000 runs of the simulation with different covariates, responses, and parameters passed to the scoring functions. The ultimate winner was distributing α by the weighting of the absolute value of the modified contribution, so that is the method we used in our final equation for mSHAP values.

11 Final mSHAP Equation Slide

Here we have the final algorithm with the absolute value-based distribution of α.

Ultimately, it allows us to write the prediction of our two-part model as the sum of a baseline term and the mSHAP values for each predictor, thus maintaining the local accuracy property.

This algorithm for computing mSHAP values for two-part models has been implemented in R and can be accessed in the mSHAP package on CRAN. Given the SHAP values and baseline terms for the parts of a two-part model we can compute the SHAP values for the final prediction!

But is it valid? Our study shows that it is.

12 Comparison to kernelSHAP slide

In addition to the simulation to distribute α, we also simulated a comparison of TreeSHAP and mSHAP. Both these were scored against kernelSHAP, which is represented on this plot as the dashed line at a perfect score of 3. TreeSHAP values cannot be computed on two-part models, but we simulated data for a single model with a target response that was equal to the two-part model's multiplied response. This simulation used the same scoring system against kernelSHAP, and spanned a wide range of possible output transformations. On the chart, we see the results of this simulation across a variety of number of predictors, with the mSHAP average score trend in red and the TreeSHAP average score trend in blue.

Recall that kernelSHAP values are approximations while TreeSHAP values are exact, leading to the divergence between kernelSHAP and TreeSHAP. However, the similarity in scores between mSHAP and TreeSHAP over a large range of covariates leads us to believe that mSHAP is a valid computation for SHAP values of two-part models.

In addition to being accurate, mSHAP is also much faster than kernelSHAP since it builds upon TreeSHAP.

12.1 ***SLIDE***

As previously mentioned, one of the biggest downfalls of kernelSHAP is the computational cost associated with it. These charts show the time comparison between mSHAP and kernelSHAP computations as the number of variables increase and as the sample size increases. Also note that these kernelSHAP values were computed with only 100 background samples, and adding more drastically slows the process.

mSHAP improves so much on the speed of kernelSHAP due to the TreeSHAP algorithm which allows for tree-based two-part models to be rapidly explained. The mSHAP time displayed in these plots includes the time taken to compute the TreeSHAP values for both model parts.

In a practical application, we had a data set of 5,000,000 observations and 45 variables that we wanted to explain. Based on the time per observation at 45 variables, we computed that kernelSHAP would take 131 days of computer-time to explain all the observations, whereas mSHAP only took 3 hours.

13 Application Slide

The practical application previously mentioned was done using a property damage insurance data set, and the steps alternated between Python and R in order to encourage interoperability between the languages. We cleaned the data in R, trained both models and calculated TreeSHAP values in python, and then passed those SHAP values back over to R to compute the mSHAP values of the final two-part model.

13.1 ***SLIDE***

Once we computed the mSHAP values on the two-part model, we could visualize the distribution of SHAP values for the 10 most important variables. These points are colored by the variable value and give us a high-level overview of the model. This plot allows us to speak in general terms about what variables are important in the model and how they contribute to predictions on average.

13.2 ***SLIDE***

We can also isolate a single prediction and visualize what the most important variable contributions are to that prediction. This plot shows how the model went from the baseline term to the ultimate prediction. We have a step-by-step guide of how the variables contributed to the final model prediction. It enables actuaries and regulators to ensure the models are fair, and gives consumers an explanation of their insurance rates. This plot gives us an explainable two-part model.

14 Conclusion Slide

In conclusion, kernelSHAP is unable to feasibly explain model predictions for two-part models, but mSHAP provides a solution to this problem. Using TreeSHAP values which can be computed rapidly, the mSHAP algorithm explains the predictions of two-part tree-based models. This advance in explainable machine learning will allow for tree-based models to become more prevalent in regulated industries such as insurance.

15 Acknowledgment Slide

I would like to thank first and foremost my mentor and fellow researcher Brian Hartman who worked on and supported this project the entire time. Also the CAS for their grant and the Statistics Department at BYU for the use of their servers.

For those interested, the paper is available on the arXiv, and the code used to create the plots/perform the simulations is available on github.

The plots I showed in the application portion of the presentation were created with the mshap R package, which is available on the CRAN and on github.

Thank you for your attention!