




ale: An R package for interpretable machine learning and statistical inference with accumulated local effects (ALE)

Chitu Okoli ¹

¹ SKEMA Business School, Paris – Université Côte d'Azur

DOI: [10.xxxxxx/draft](https://doi.org/10.xxxxxx/draft)

Software

- [Review](#) 
- [Repository](#) 
- [Archive](#) 

Editor: [Susan Holmes](#) 

Reviewers:

- [@martinju](#)

Submitted: 28 April 2025

Published: unpublished

License

Authors of papers retain copyright and release the work under a Creative Commons Attribution 4.0 International License ([CC BY 4.0](https://creativecommons.org/licenses/by/4.0/))

Summary

Machine learning (ML) models, particularly those referred to as “black-box” models, achieve high predictive accuracy at the cost of interpretability. Interpretable Machine Learning (IML) aims to provide tools to understand these complex models. The Accumulated Local Effects (ALE) methodology, introduced by Apley and Zhu (2020), has emerged as a powerful, model-agnostic IML technique to visualize and interpret black-box model predictions. Unlike alternative methods such as Partial Dependence Plots (PDP) or SHapley Additive exPlanations (SHAP), ALE plots clearly represent variable effects independent of interaction complexities and computationally scale better with large datasets.

The `ale` package for R provides advanced implementations of ALE, addressing existing limitations and introducing critical statistical inference tools to enhance interpretability. It offers robust bootstrapped confidence intervals for ALE plots, effect size measures on both outcome-variable and normalized scales, and intuitive visualization methods, empowering users to draw statistically sound inferences from their models. First published in 2023, its current version is 0.5.0.

Statement of Need

While ALE methods hold considerable promise, their effective application faces three main challenges:

- 1. Reliability in small datasets:** Typical machine learning techniques, including ALE, assume large datasets capable of supporting training-test splits for validation. However, statistical analyses frequently involve smaller datasets (fewer than 2,000 observations), posing risks of overfitting and limiting the generalizability of results. The `ale` package implements specialized bootstrapped confidence intervals adapted for small datasets, ensuring that ALE results remain reliable across diverse research contexts.
- 2. Intuitive characterization of variable effects:** Existing IML methods often lack intuitive, interpretable effect-size metrics that summarize the overall impact of predictor variables clearly and meaningfully. The `ale` package introduces novel ALE effect size measures (ALE Deviation (ALED), ALE Range (ALER), Normalized ALE Deviation (NALED), and Normalized ALE Range (NALER)), explicitly designed for ease of interpretation and comparability across different datasets and models.
- 3. Robust statistical inference with ML:** Traditional statistical inference typically relies on parametric assumptions and linear models. However, ML models frequently violate these assumptions due to their inherent flexibility. By integrating rigorous bootstrapping methods and effect-size-based inference mechanisms, `ale` bridges ML flexibility with

the rigor of classical statistical methods. The package clearly delineates statistically significant effects, distinguishing meaningful relationships from random variations via carefully constructed confidence intervals and p-value distributions.

Software implementations of ALE

In the following table, we list some key characteristics of alternative software implementations of ALE, contrasting them with some of the key unique features of the ale package.

Software packages that implement ALE

Pri- mary focus	Package	Latest release	Lan- guage	Confi- dence intervals	Boot- strap type	ALE statistics
ALE	ALEPlot (Apley, 2018)	2018	R	No	N/A	None
ALE	ALEPython (Jumelle et al., 2020)	2020	Python	Monte Carlo	data-only	None
IML	iml (Molnar & Schratz, 2022)	2025	R	No	N/A	None
IML	DALEX (Biecek et al., 2023)	2023	R and Python	No	N/A	None
ALE	PyALE (Jomar, 2023)	2024	Python	T-statistic	N/A	None
IML	Interpretation (RapidMiner, 2023)	2024	Rapid-Miner	No	N/A	None
IML	Alibi (Seldon Technologies, 2023)	2024	Python	No	N/A	None
IML	scikit-explain (Flora, 2023)	2023	Python	Boot-strap	data-only	<ul style="list-style-type: none">Friedman H-statistic for interactionsInteraction strength (IAS)Main effect complexity (MEC)
ALE	ale (introduced in this article)	2025	R	Boot-strap	data-only and model	<ul style="list-style-type: none">ALE deviation (ALED)ALE range (ALER)Normalized ALED (NALED)Normalized ALER (NALER)

47 Simple demonstration

48 To demonstrate some capabilities of the package, we will give two demonstrations: first, a
49 simple demonstration of ALE plots, and second, a more sophisticated demonstration suitable
50 for statistical inference with p-values. For both demonstrations, we begin by fitting a GAM
51 model. We assume that this is a final deployment model that needs to be fitted to the entire
52 dataset.

```
library(ale)
```

53 Attaching package: 'ale'

54

55 The following object is masked from 'package:base':

56

57 get

```
# Sample 1000 rows from the ggplot2::diamonds dataset (for a simple example).  
set.seed(0)
```

```
diamonds_sample <- ggplot2::diamonds[sample(nrow(ggplot2::diamonds), 1000), ]
```

```
# Create a GAM model with flexible curves to predict diamond price.  
# Smooth all numeric variables and include all other variables.  
# Build model on training data, not on the full dataset.
```

```
gam_diamonds <- mgcv::gam(  
  price ~ s(carat) + s(depth) + s(table) + s(x) + s(y) + s(z) +  
    cut + color + clarity +  
    ti(carat, by = clarity), # a 2D interaction  
  data = diamonds_sample  
)
```

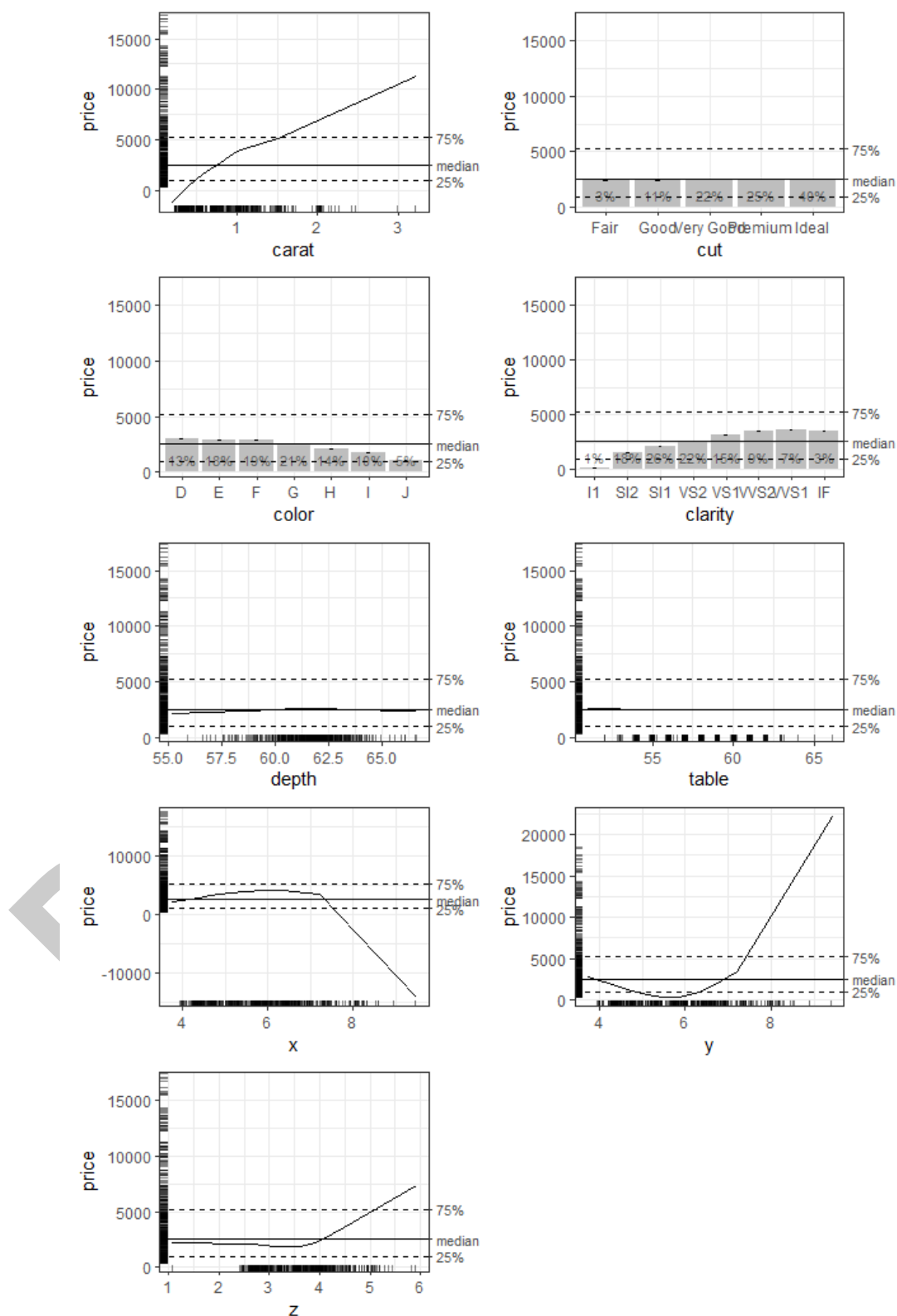
58 First, we directly create ALE data with the ALE() constructor and then plot the ggplot plot
59 objects.

```
# Create ALE data
```

```
ale_gam_diamonds <- ALE(gam_diamonds, data = diamonds_sample)
```

```
# Plot the ALE data
```

```
plot(ale_gam_diamonds) |>  
  print(ncol = 2)
```



60

61 To demonstrate the ALE statistics functionality, we need to create a p-value distribution object
62 so that the ALE statistics can be properly distinguished from random effects.

Create p_value distribution object

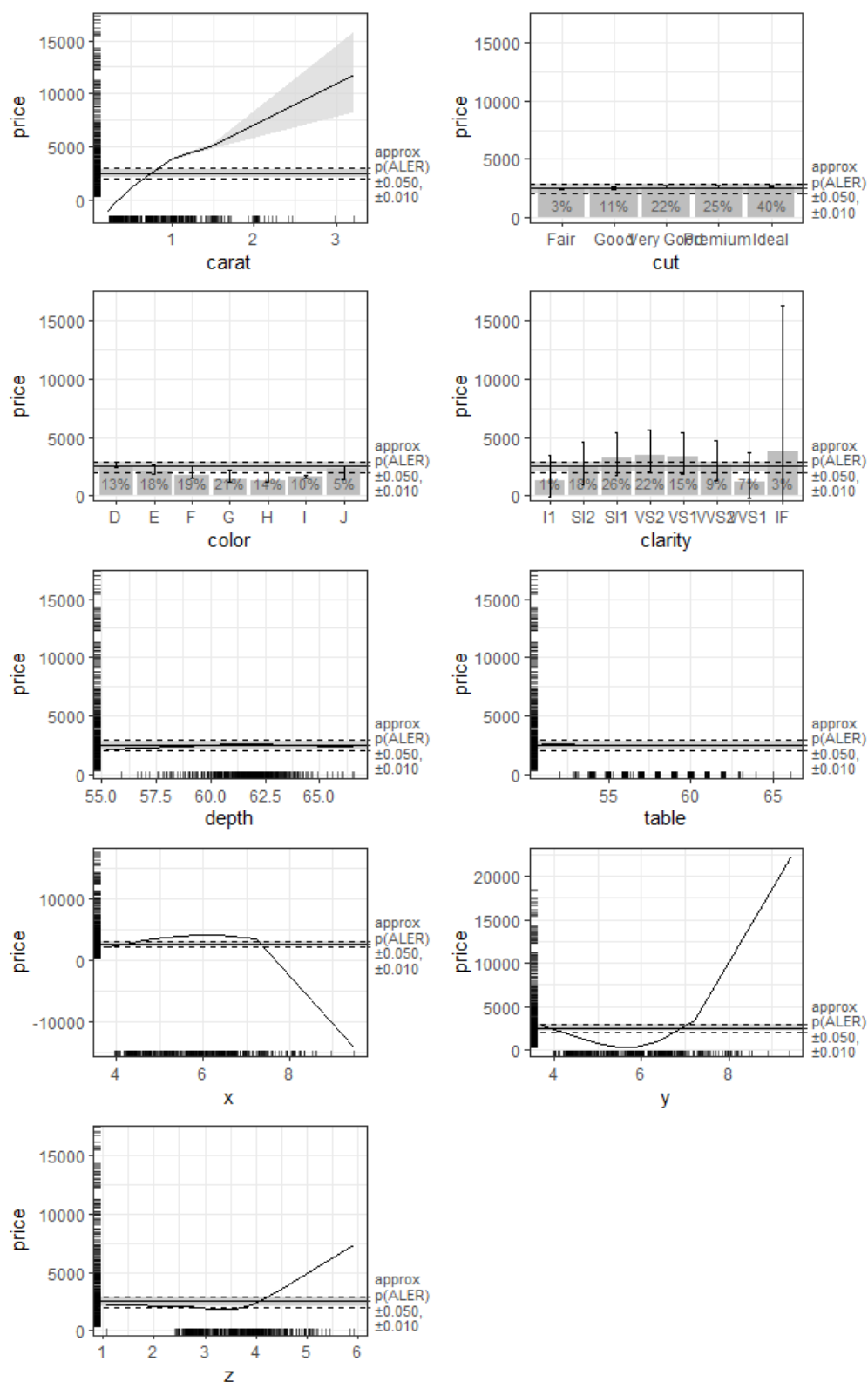
```
# # To generate the code, uncomment the following lines.
# # But it is slow because it retrains the model 100 times, so this vignette loads a pre
# gam_diamonds_p_readme <- ALEpDist(
#   gam_diamonds, diamonds_sample,
#   # Normally should be default 1000, but just 100 for quicker demo
#   rand_it = 100
# )
# saveRDS(gam_diamonds_p_readme, file.choose())
gam_diamonds_p_readme <-
  url('https://github.com/tripartio/ale/raw/main/download/gam_diamonds_p_readme.0.5.0.rds')
  readRDS()

63 Now we can create bootstrapped ALE data and see some of the differences in the plots of
64 bootstrapped ALE with p-values:

# Create ALE data
ale_gam_diamonds_stats_readme <- ALE(
  gam_diamonds,
  # generate all for all 1D variables and the carat:clarity 2D interaction
  x_cols = list(d1 = TRUE, d2 = 'carat:clarity'),
  data = diamonds_sample,
  p_values = gam_diamonds_p_readme,
  # Usually at least 100 bootstrap iterations, but just 10 here for a faster demo
  boot_it = 10
)

# Create an ALEplots object for fine-tuned plotting
ale_plots <- plot(ale_gam_diamonds_stats_readme)

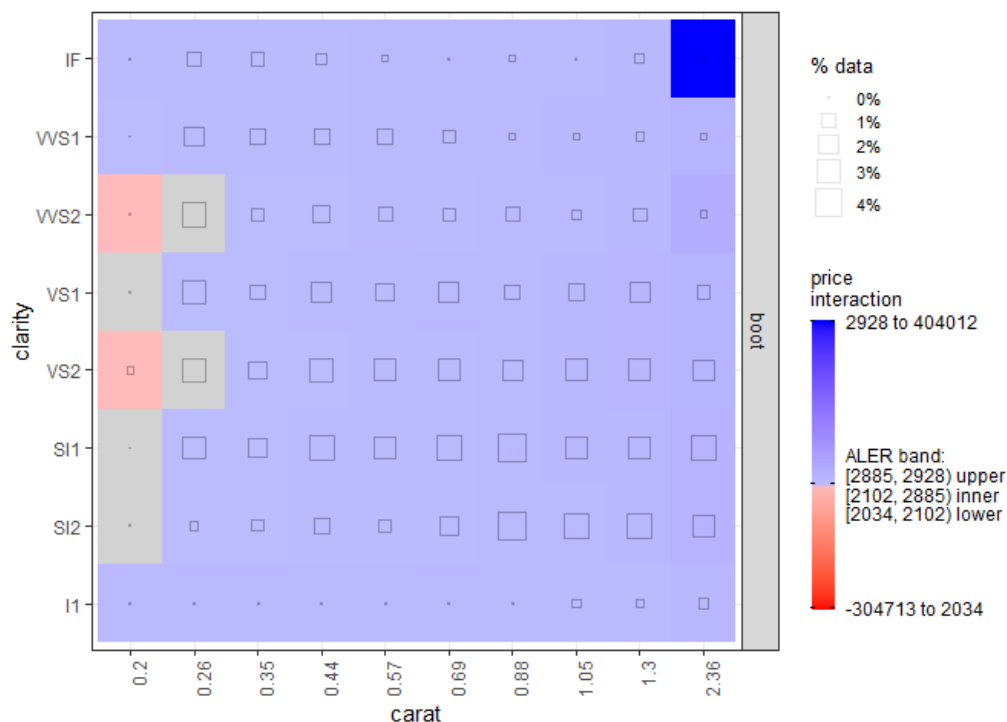
# Plot 1D ALE plots
ale_plots |>
  # Only select 1D ALE plots.
  # Use subset() instead of get() to keep the special ALEplots object
  # plot and print functionality.
  subset(list(d1 = TRUE)) |>
  print(ncol = 2)
```



65

```
# Plot a selected 2D plot
ale_plots |>
  # get() retrieves a specific desired plot
```

```
get('carat:clarity')
```



Scientific basis

The scientific basis of the `ale` package is best described in a working paper that describes ALE statistics in detail and introduces ALE-based inference (Okoli 2023). This research has been presented at Okoli (2024) and Okoli (2024-10-20/2024-10-23).

References

- Apley, Daniel W. 2018. "ALEPlot: Accumulated Local Effects (ALE) Plots and Partial Dependence (PD) Plots." <https://doi.org/10.32614/cran.package.aleplot>.
- Apley, Daniel W., and Jingyu Zhu. 2020. "Visualizing the Effects of Predictor Variables in Black Box Supervised Learning Models." *Journal of the Royal Statistical Society Series B: Statistical Methodology* 82 (4): 1059–86. <https://doi.org/10.1111/rssb.12377>.
- Biecek, Przemyslaw, Szymon Maksymiuk, and Hubert Baniecki. 2023. "DALEX: moDEL Agnostic Language for Exploration and eXplanation." <https://doi.org/10.32614/cran.package.dalex>.
- Flora, Montgomery. 2023. "Scikit-Explain: A User-Friendly Python Package for Computing and Plotting Machine Learning Explainability Output."
- Jomar, Dana. 2023. "PyALE: ALE Plots with Python."
- Jumelle, Maxime, Alexander Kuhn-Regnier, and Sanjif Rajaratnam. 2020. "ALEPython." Blent.ai.
- Molnar, Christoph, and Patrick Schratz. 2022. "Iml: Interpretable Machine Learning."
- Okoli, Chitu. 2024-10-20/2024-10-23. "Reliable Inference from Human-Centred Datasets with Accumulated Local Effects." In *2024 INFORMS Annual Meeting*. Seattle, USA: INFORMS.

- 88 ———. 2023. "Statistical Inference Using Machine Learning and Classical Techniques Based
89 on Accumulated Local Effects (ALE)." arXiv. <https://doi.org/10.48550/arXiv.2310.09877>.
- 90 ———. 2024. "Model-Agnostic Interpretability: Effect Size Measures from Accumulated Local
91 Effects (ALE)." In *INFORMS Workshop on Data Science 2024*. Seattle, USA: INFORMS.
- 92 RapidMiner. 2023. "Interpretation." RapidMiner.
- 93 Seldon Technologies. 2023. "Alibi: Algorithms for Monitoring and Explaining Machine Learning
94 Models."
- 95 Apley, D. W. (2018). *ALEPlot: Accumulated Local Effects (ALE) Plots and Partial Dependence
96 (PD) Plots*. <https://doi.org/10.32614/cran.package.aleplot>
- 97 Biecek, P., Maksymiuk, S., & Baniecki, H. (2023). *DALEX: moDel Agnostic Language for
98 Exploration and eXplanation*. <https://doi.org/10.32614/cran.package.dalex>
- 99 Flora, M. (2023). *Scikit-explain: A user-friendly python package for computing and plotting
100 machine learning explainability output*.
- 101 Jomar, D. (2023). *PyALE: ALE plots with python*.
- 102 Jumelle, M., Kuhn-Regnier, A., & Rajaratnam, S. (2020). *ALEPython*. Blent.ai.
- 103 Molnar, C., & Schratz, P. (2022). *Iml: Interpretable Machine Learning*.
- 104 RapidMiner. (2023). *Interpretation*. RapidMiner.
- 105 Seldon Technologies. (2023). *Alibi: Algorithms for monitoring and explaining machine learning
106 models*.