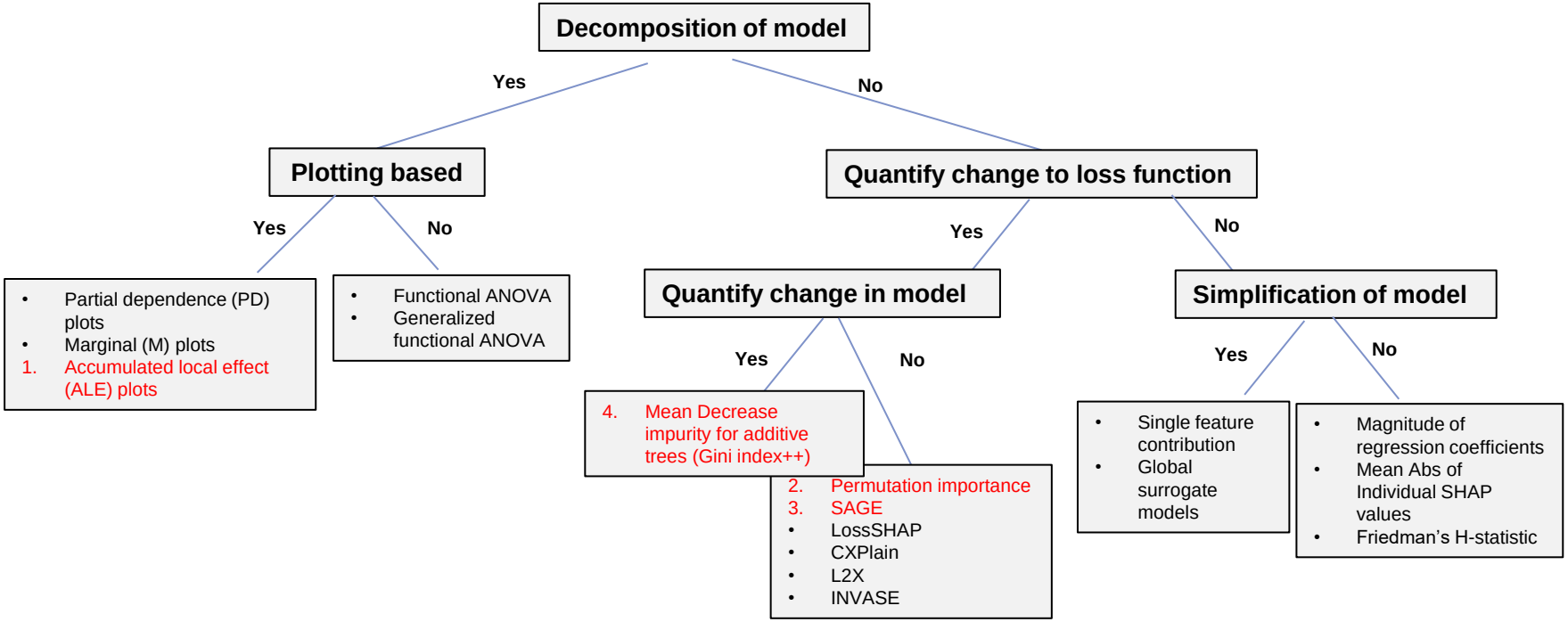
Norsk Regnesentral
NORWEGIAN COMPUTING CENTER

Global model-agnostic* explanation methods

Martin Jullum and Annabelle Redelmeier

April 8th, 2022

Overview of global explanation methods



A closer look at 4 methods

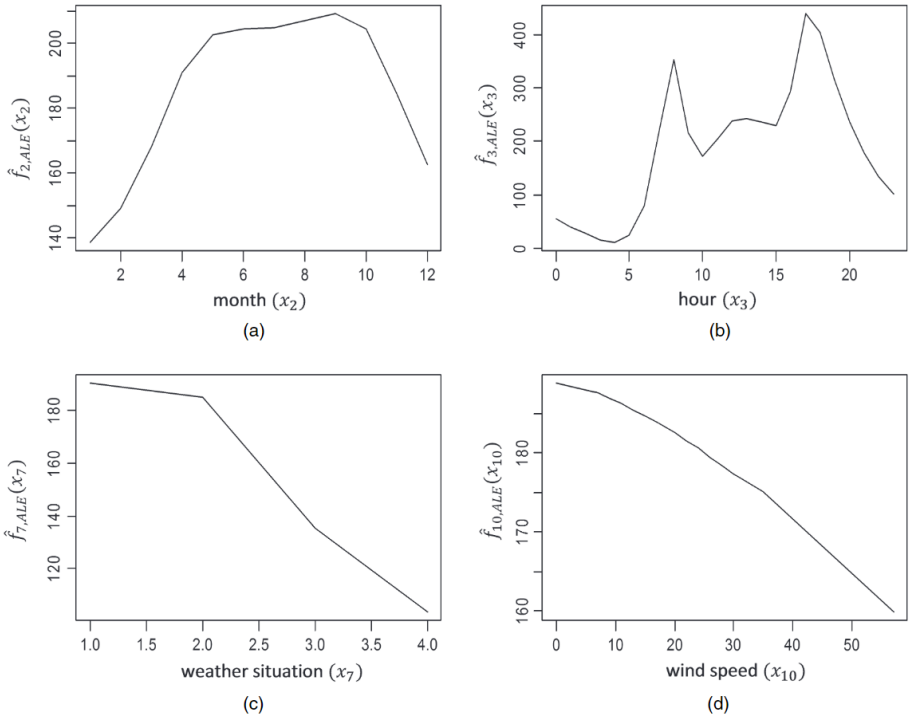
1. *ALE Plots*
   - **Visualizing the effects of predictor variables in black box supervised learning models** by Daniel W. Apley and Jingyu Zhu, 2020
2. *Permutation Feature Importance*
   - **Random Forests** by Breiman, 2001
   - **All models are wrong, but many are useful: Learning a variable's importance by studying an entire class of prediction models simultaneously** by Aaron Fisher, Cynthia Rudin, and Francesca Dominici, 2018
3. *SAGE*
   - **Understanding global feature contributions with additive importance measures** by Ian Covert, Scott Lundberg, and Su-In Lee, 2021
4. *Mean decrease impurity for additive trees*
   - **Understanding variable importances in forests of randomized trees**, Louppe et al. (2013)
   - **Elements of Statistical Learning,** Ch 10.13, Hastie et al. (2001)
   - **Classification and Regression Trees,** Ch 4+5, Breiman et al. (1984)

(1) Accumulated local affects (ALE) plots

Apley and Zhu, 2020

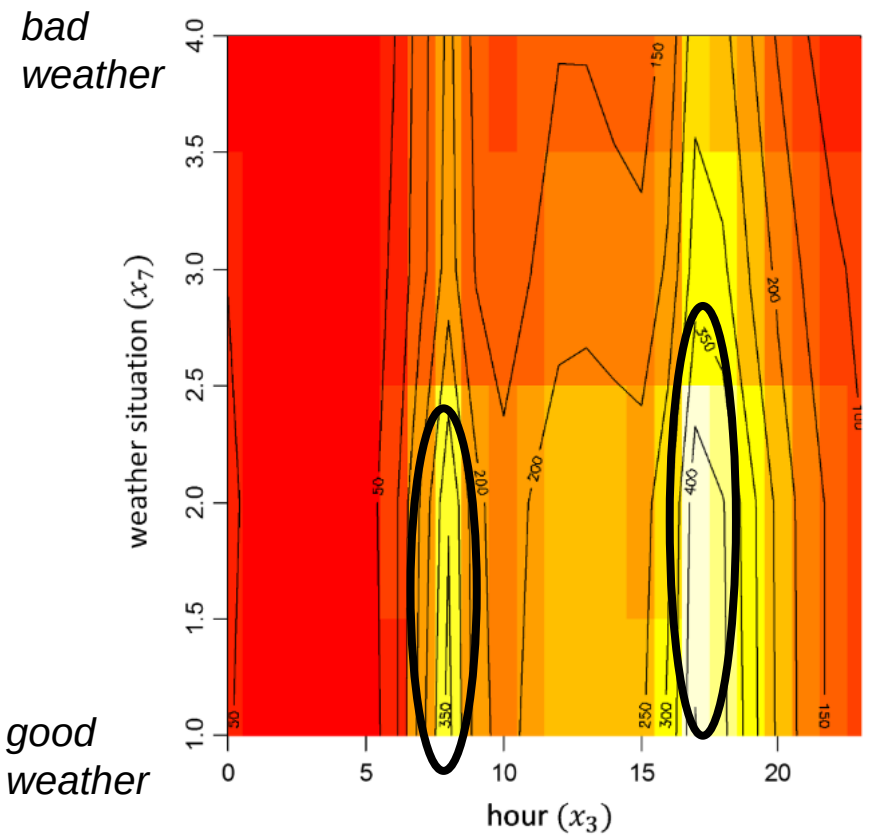
- Imagine a bike rental problem where ***y = # bike rentals per hour.***

One sentence explainer: The ALE function value for a given feature is the ***predicted response as a function of*** X_i, when all other features are averaged out.



2nd order ALE plots

- The higher the peaks, the more ***hour*** and ***weather situation*** have an influence on # bike rentals.
- If good weather, the bike rental peaks are pronounced in the morning and evening rush hour.
- If bad weather the peaks are at the same time (but less pronounced)



Inspiration for ALE comes from PD and M plots

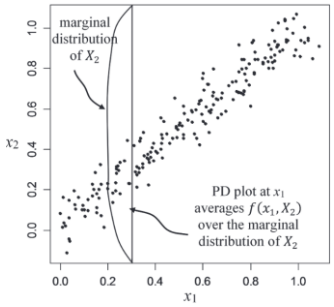
The PD function of X_1 shows the marginal effect X_1 has on the predicted outcome of the model.

The M function fixes the extrapolating problem by replacing marginal w/ conditional dist.

$$f_{1,\mathrm{PD}}(x_1) \equiv \mathbb{E}[f(x_1, X_2)] = \int p_2(x_2)\, f(x_1, x_2)\mathrm{d}x_2 \qquad f_{1,\mathrm{M}}(x_1) \equiv \mathbb{E}[f(X_1, X_2)|X_1 = x_1] = \int p_{2|1}(x_2|x_1)\, f(x_1, x_2)\mathrm{d}x_2$$

In practice:

1. Divide X_1 into n segments.
2. For each segment, calculate avg model prediction over the ***marginal distribution*** of X_2

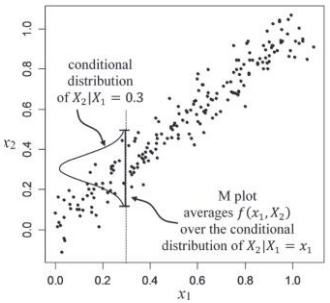


Problem?

Bad at extrapolating

In practice:

1. Divide X_1 into n segments.
2. For each segment, calculate avg model prediction over the ***conditional distribution*** of X_2



Problem?

Estimate combined effect

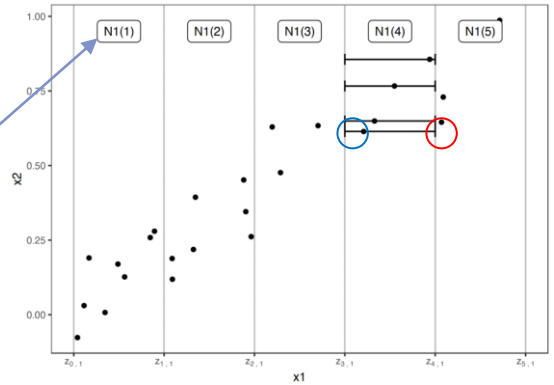
Inspiration for ALE comes from PD and M plots

The ALE function fixes the conflation issue by taking **differences** $f(z_{1,upper}, x_2) - f(z_{1,lower}, x_2)$

$$f_{1,\mathrm{ALE}}(x_1) \equiv \int_{x_{\min,1}}^{x_1} \mathbb{E}[f^1(X_1, X_2)|X_1 = z_1]\,\mathrm{d}z_1 - \text{constant}$$

In practice:

1. Divide X_1 into n segments.
2. For each segment, calculate avg **local affect** $f(z_{1,upper}, x_2) - f(z_{1,lower}, x_2)$
3. Take cumsum from N1(1) to N1(i).



Summary: ALE plots

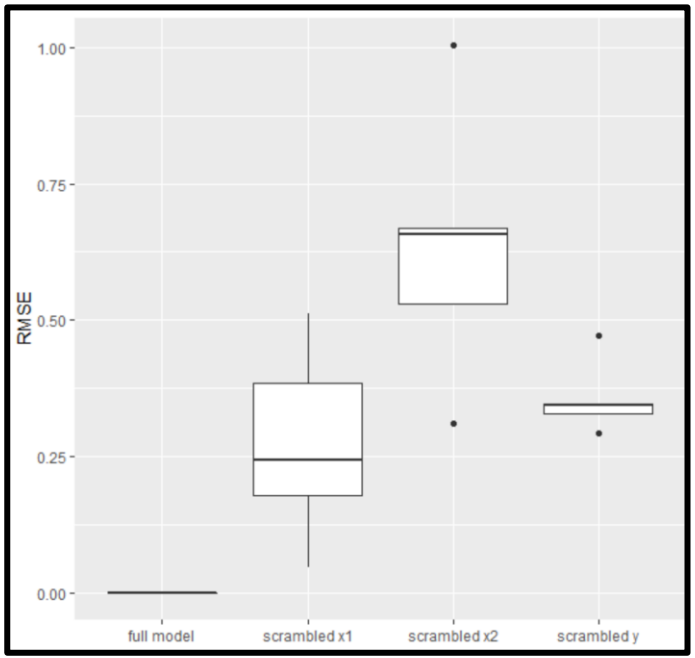
- Estimate the average prediction value for a given feature (or two features) value.
  - Diagnose obvious relationship problems b/w Y and feature
- Advantages
  - **ALE plots handle dependent features.**
  - **ALE plots are faster to compute** than PD plots.
- Disadvantages
  - Second-order ALE estimates have **a varying stability** across the feature space which are not visualized.
  - Second-order effect plots can be a bit hard to interpret, as you **always have to keep the main effects in mind**. It is tempting to read the heat maps as the total effect of the two features, but it is only the additional effect of the interaction.

(2) Permutation Feature Importance

Breiman, 2001

One sentence explainer: A feature's importance is tied to how model's error changes when the feature's information is destroyed.

When X_1 info destroyed:
model error ↑ : X_1 important
model error ↓/same: X_1 not important

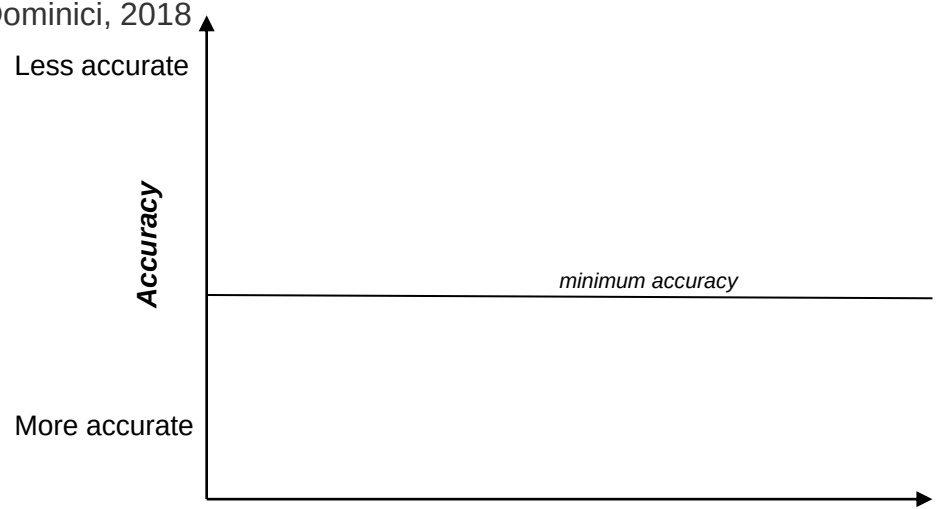


In practice: *feature importance of* X_1

1. Calculate model loss: $Loss(X)$
2. Randomly set $X_{i,1} = X_{j,1}$ for $i = 1, \dots, n$
3. Calculate $Loss(scrambled\ X)$
4. Calculate $\frac{Loss(scrambled\ X)}{Loss(X)}$ or $Loss(scrambled\ X) - Loss(X)$

Robust Permutation Feature Importance

Fisher, Rudin, and Dominici, 2018



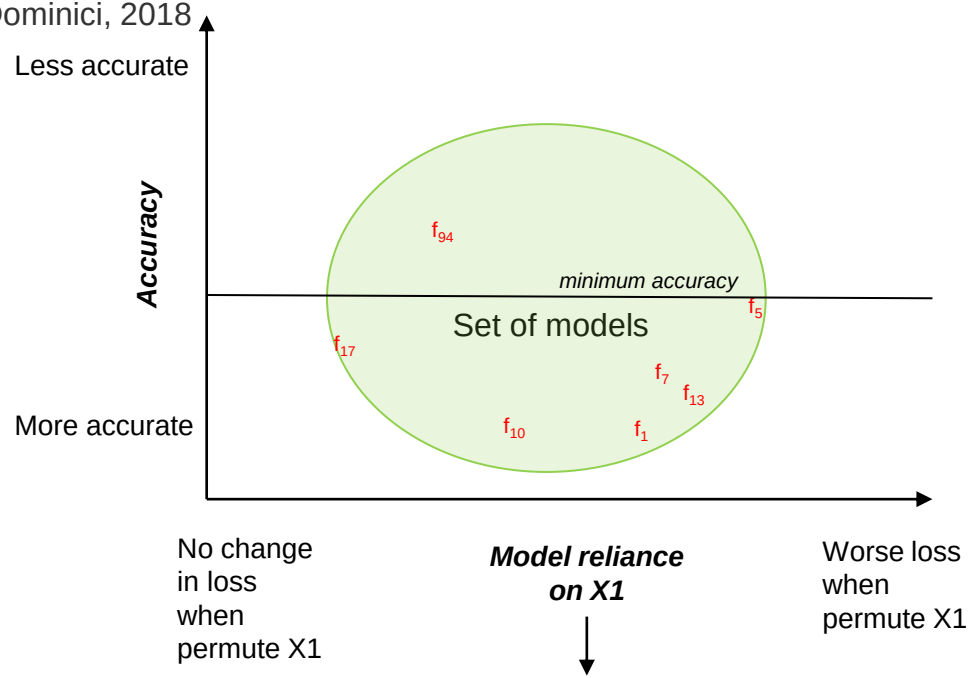
Set of models

f_{13} f_5 f_{17} f_{10} f_1 f_7 f_{94}

$$\frac{Loss(scrambled\ X)}{Loss(X)}$$

Robust Permutation Feature Importance

Fisher, Rudin, and Dominici, 2018



$$\frac{Loss(scrambled\ X)}{Loss(X)}$$

How large is this spread?

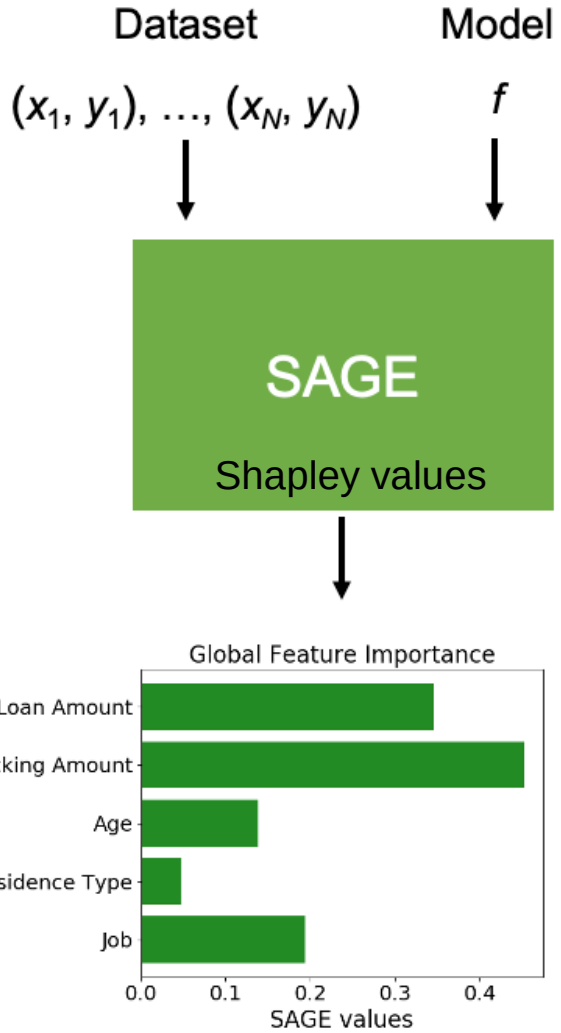
Summary: Permutation feature importance

- The increase in model error when the feature's information is destroyed.
- Advantages
  - The importance measure automatically **takes into account all interactions** with other features
    - Also a disadvantage because the importance of the interaction between two features is included in the importance measurements of both features.
  - No **retraining** of the model.
- Disadvantages
  - Linked to a **specific choice of *error*** of the model.
  - You **need access to the true outcome**.
  - Permutations are random.
  - If features are correlated, PFI **can be biased by unrealistic data instances**.
  - Adding a **correlated feature can decrease the importance of the associated feature** by splitting the importance between both features.

(3): SAGE
(Shapley Additive Global importancE)

- Covert, Lundberg & Lee (NeurIPS, 2020)
- Using Shapley values to decompose the expected loss of the model on the features

One sentence explainer: How much the expected loss is reduced by including each of the features to the model (averaged over whether the other features are included or not)



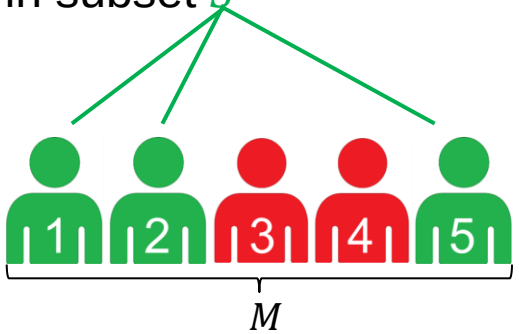
Shapley values

- Concept from (cooperative) game theory in the 1950s
- Used to distribute the total payoff to the players
- Explicit formula for the “fair” payment to every player j:

$$\phi_j = \sum_{S \subseteq M \setminus \{j\}} \frac{|S|!\,(|M| - |S| - 1)!}{|M|!} \left(v(S \cup \{j\}) - v(S)\right)$$

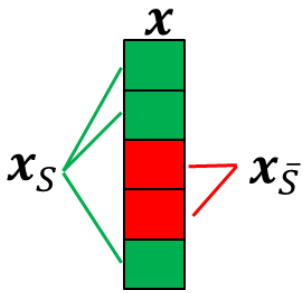
$v(S)$ is the payoff with only players in subset S

- Several mathematical optimality properties



$$\phi_j = \sum_{S \subseteq M \setminus \{j\}} \frac{|S|!\,(|M| - |S| - 1)!}{|M|!} \left(v(S \cup \{j\}) - v(S)\right)$$

Shapley value explanations



- Individual prediction explanation (local)
  - SHAP: Popularised by Lundberg & Lee (2017)
    - Players = features $(x_1, \ldots, x_M)$
    - Payoff = difference between prediction to mean prediction $f(\boldsymbol{x}^*) - \mathrm{E}_{\mathbf{X}}[f(\boldsymbol{X})]$
    - Contribution function: $v(S) = v_{l,S}(\boldsymbol{x}_S^*) = \mathrm{E}_{\boldsymbol{X}_{\bar{S}}}[f(\boldsymbol{X})|\boldsymbol{X}_S = \boldsymbol{x}_S^*]$
    - Marginal contributions: $v(S \cup \{j\}) - v(S) = \mathrm{E}_{\boldsymbol{X}_{\overline{S \cup j}}}\left[f(\boldsymbol{X})\middle|\boldsymbol{X}_{S \cup j} = \boldsymbol{x}_{S \cup j}^*\right] - \mathrm{E}_{\boldsymbol{X}_{\bar{S}}}[f(\boldsymbol{X})|\boldsymbol{X}_S = \boldsymbol{x}_S^*]$

- Whole model explanation (global)
  - SAGE: Covert, Lundberg & Lee (2020)
    - Players = features $(x_1, \ldots, x_M)$
    - Payoff = Difference between expected loss with constant model ($f_c(x) = \mathrm{c} = \mathrm{E}_{\mathbf{X}}[f(\boldsymbol{X})]$) and full model: $\mathrm{E}_Y[l(c, Y)] - \mathrm{E}_{\boldsymbol{X},Y}[l(f(\boldsymbol{X}), Y)]$
    - Contribution function: $v(S) = \mathrm{E}_Y[l(c, Y)] - \mathrm{E}_{\boldsymbol{X}_S,Y}[l(v_{l,S}(\boldsymbol{X}_S), Y)]$
    - Marginal contributions: $v(S \cup \{j\}) - v(S) = \mathrm{E}_{\boldsymbol{X}_S,Y}[l(v_{l,S}(\boldsymbol{X}_S), Y)] - \mathrm{E}_{\boldsymbol{X}_{S \cup j},Y}[l(v_{l,S \cup j}(\boldsymbol{X}_{S \cup j}), Y)]$

Estimating the expectations in SAGE

- Need to estimate

$v(S \cup \{j\}) - v(S) = \mathrm{E}_{\boldsymbol{X_S},Y}[l(v_{l,S}(\boldsymbol{X}_S), Y)]$ - $\mathrm{E}_{\boldsymbol{X_{S\cup j}},Y}[l(v_{l,S\cup j}(\boldsymbol{X}_{S\cup j}), Y)]$

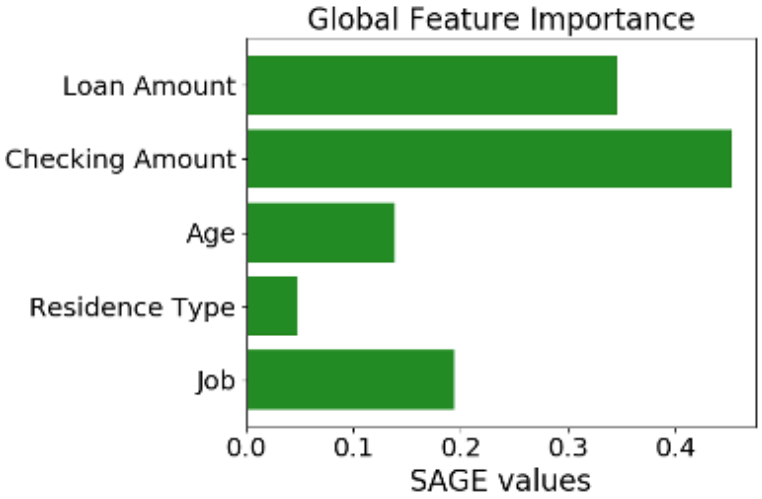
  - Estimate the $v_{l,S}(\boldsymbol{x}_S) = \mathrm{E}_{\boldsymbol{X}_{\bar{S}}}[f(\boldsymbol{X})|\boldsymbol{X}_S = \boldsymbol{x}_S]$ using $\mathrm{E}_{\boldsymbol{X}_{\bar{S}}}[f(\boldsymbol{X}_{\bar{S}}, \boldsymbol{x}_{\bar{S}})]$ and approximate it by sampling rows of $\boldsymbol{X}_{\bar{S}}$ from the training set
  - Sample from the training set to estimate the outer expectation $\mathrm{E}_{\boldsymbol{X_S},Y}$ by sampling full rows $\boldsymbol{X}, Y$ from the training set

- Algorithm for computing the Shapley values is based on previous work by Strumbelj & Kononenko (2010)

Properties of SAGE

- ϕ_j = Reduction in expected loss caused by including feature j in the model (averaged over whether the other features are included or not)

- $\sum_j \phi_j = \mathrm{E}_Y[l(c, Y)] - \mathrm{E}_{\boldsymbol{X},Y}[l(f(\boldsymbol{X}), Y)]$
- $\phi_j = 0$ => No change in model performance change by feature j



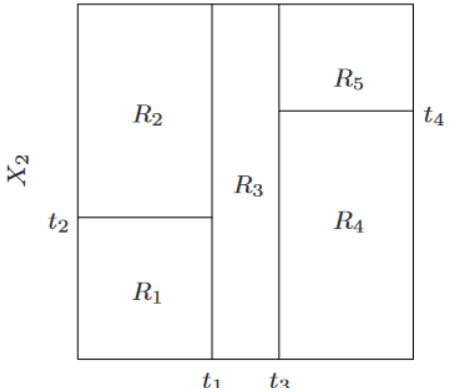
“Similar” methods

- LossSHAP
  - A local explanation method which decomposes $l(f(\boldsymbol{x}^*), y)$ for a given $\boldsymbol{x}^*$and y, instead of the usual $f(\boldsymbol{x}^*)$
  - SAGE is the mean of lossSHAP over the dataset
    - Computing global explanations with SAGE directly is faster as we don’t need to compute precise lossSHAP values for every pair $(\boldsymbol{x}^*, y)$.
    - IMO they oversell their SAGE-algorithm as what they do is essentially to compute lossSHAP for sampled data using a single S.
- Feature ablation and permutation features
  - Also look at differences in expected loss by simulating removal of a feature like SAGE, but they don’t consider multiple feature subsets through Shapley values – only consider removal from the full model
  - Feature ablition
    - Simulates removal of a feature by re-training the model
  - (One version of) permutation features
    - Simulates removal of a feature by permuting the input value

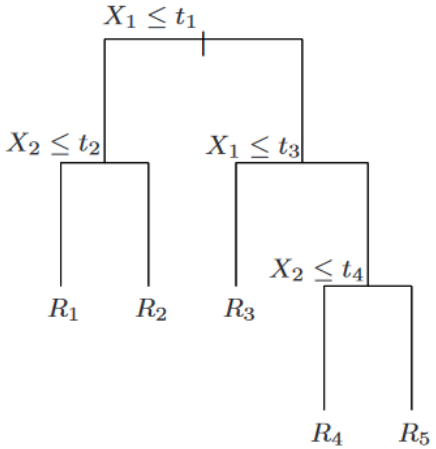
Summary SAGE

- Uses Shapley values to decompose the expected loss of the model onto the features
- Advantages
  - Theoretical foundation
  - Generalizes several other methods
- Disadvantages
  - Author's implementation (*shap* in pyhton) does not account for feature dependence
  - Computationally costly (at least if accounting for dependence)

(4) Mean decrease impurity (MDI) for additive trees



- Consider a trained tree model
  - Each split aims at minimizing a loss/«impurity» measure
  - Importance for feature j = weighted sum of decrease in impurity due to a split in feature j
  - Importance scores typically scaled to sum to 1
- Random forest and boosted trees
  - Similar to single trees, but sums over all trees before scaling



One sentence explainer: What proportion of the performance increase is due to splits in the different features

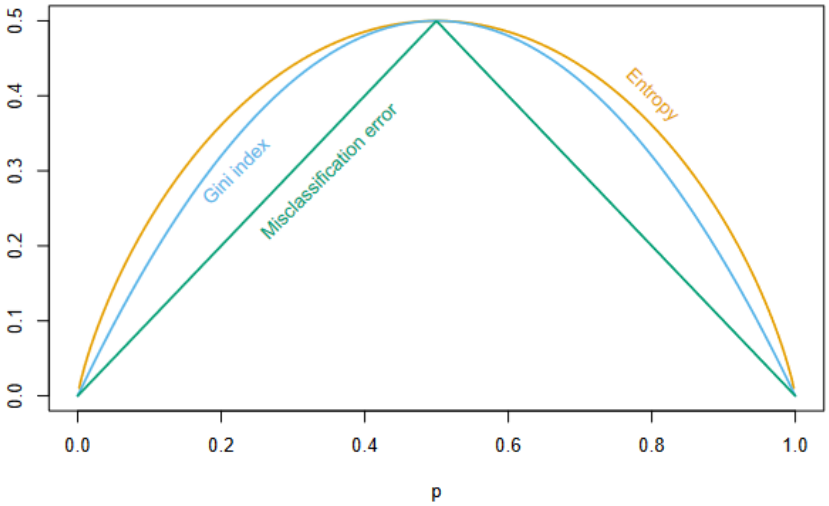
MDI mathematical definition

- Impurity decrease at node t $\Delta i(s,t) = i(t) - p_L i(t_L) - p_R i(t_R)$
  - p_L, p_R are the proportion of samples in the left and right split
- Importance of feature X_m: $Imp(X_m) = \frac{1}{N_T} \sum_T \sum_{t \in T: v(s_t) = X_m} p(t) \Delta i(s_t, t)$
  - $p(t)$ is the proportion of samples reaching node t
- Typically $Imp(X_m)^* = \frac{Imp(X_j)}{\sum_j Imp(X_j)}$

NR

MANY variations: impurity measures

- Impurity measure used to perform splits in tree models/random forest
  - Classification: Missclassification error, Gini index or cross-entropy
  - Regression: MSE
- Boosted trees algorithms (xgboost, lightgbm, catboost etc) splits in other ways, often an approximation to some given loss
  - Importance typically defined based on the approximated loss



Misclassification error: $\frac{1}{N_m}\sum_{i\in R_m} I(y_i \neq k(m)) = 1 - \hat{p}_{mk(m)}.$

Gini index: $\sum_{k\neq k'} \hat{p}_{mk}\hat{p}_{mk'} = \sum_{k=1}^{K} \hat{p}_{mk}(1-\hat{p}_{mk}).$

Cross-entropy or deviance: $-\sum_{k=1}^{K} \hat{p}_{mk}\log\hat{p}_{mk}.$

MANY variations: scaling

- ESL Ch 10.13 and the Xgboost implementation (total gain)
  - does not seem to include the $p(t)$ to weight the performance increase by their position in the tree
- ESL suggest using the squared impurity decrease instead $\Delta i^2(s,t)$, taking the sum over the trees, and then the square root before scaling to 1

MANY variations: Close relatives

- Breiman (1984)'s original idea for single tree
  - Measure improvement in surrogate split instead to avoid problem with "masked features"
    - Masking is less of an issue for random forest/boosting
- XGBoost's «average gain»
  - Average performance increase when feature j is used (instead of total performance increase)
- Mean Decrease accuracy (MDA) for random forest:
  - Measure performance increase in out-of-bag-samples

Summary MDI

- MDI assigns the performance increase by every split to the feature performing the split
- Interpretation of MDI_j: Proportion of the model's total performance increase which is due to feature j
- Advantages
  - Works for and is available in almost all tree-based modelling implementations
- Disadvantages
  - Dependence between features is only accounted through the random sampling in the trees – no importance is shared between dependent features
  - Lots of variations, difficult to know exactly what is implemented
  - Not a model-agnostic method
  - Theoretical properties not well studied
  - Importance biased towards high-cardinality features

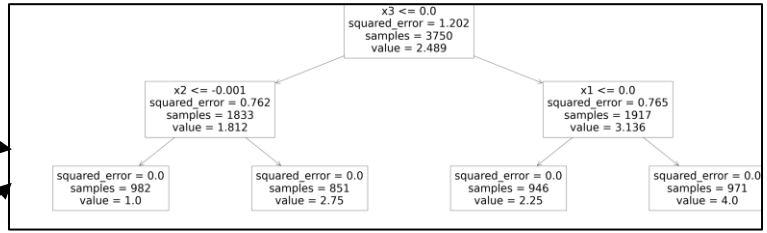
Comparing methods

What we get and what we miss!

	ALE plots	Permutation feature importance (PFI)	SAGE	Mean decrease impurity (MDI)
What the explanation tells us	• **How the fitted model prediction is changed as one or two features are changed:** Accounts for effects of other features by averaging them out	• **How much the training performance decreases if we did not observe certain features:** Simulates dropping one feature at a time	• **How much the training performance decreases if we did not observe certain features:** Accounts for and averages over whether other features are observed or not.	• **How central the different features are to reach a good fit this specific fitted model.**
What the explanation does not include	• **Joint effects of many features**: Pairwise effects OK, but more difficult to visualize • **Stability of the effects:** How much the effect varies with the features that are averaged-out	• **Importance shared with other features:** Only one feature is permuted at once, keeping the rest fixed • **Dependence awareness**: Standard version permutes features independently	• **Dependence awareness:** When measuring expected changes in performance, dependence is ignored (in implementation) • **Exact answer:** Approximations are required, especially in high-dimensions	• **Indirect importance:** The importance is not shared among highly dependent features unless the model put's equal weight on them • **Importance for non-tree-models:** The method only works for tree-based models.

Simulations

- $X_1 = X_2 \sim$ Uniform(0, 1) + Normal(0, 0.05)
  1. Linear model: $Y = X_1 + X_2^2$
- $(X_1, X_2, X_3) \sim$ Normal(0, *low corr*)
  2. Linear model: $Y = X_1 + X_2 + 2X_3$
  3. Tree model $Y = \text{tree}(X_1, X_2, X_3)$
- $(X_1, X_2, X_3) \sim$ Normal(0, *high corr*)
  4. Linear model: $Y = X_1 + X_2 + 2X_3$
  5. Tree model $Y = \text{tree}(X_1, X_2, X_3)$



How do global explanations change with different models and feature dependence?

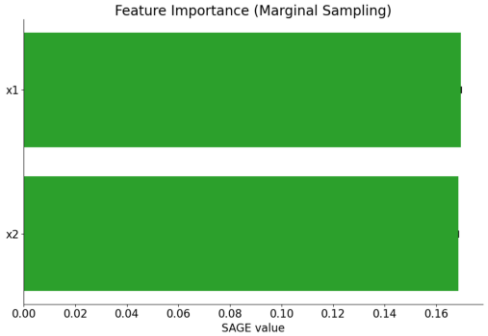
Simulations

True **linear** model: $Y = X_1 + X_2^2$

$X_1 = X_2 \sim$ Uniform(0, 1) + Normal(0, 0.05)

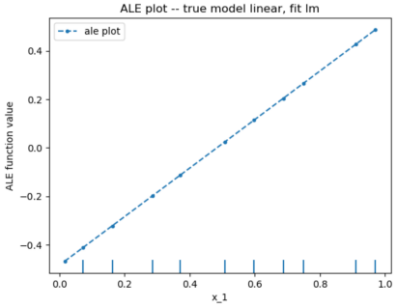
```
Permutation importance:
x2      0.511 +/- 0.026
x1      0.476 +/- 0.024
```

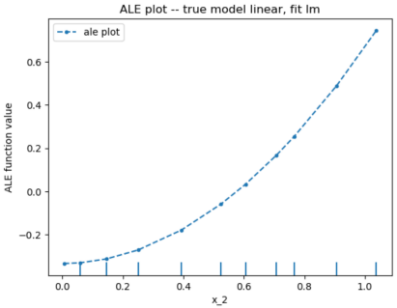
Equation contribution from both x1 and x2



Almost identical SAGE values

ALE plots





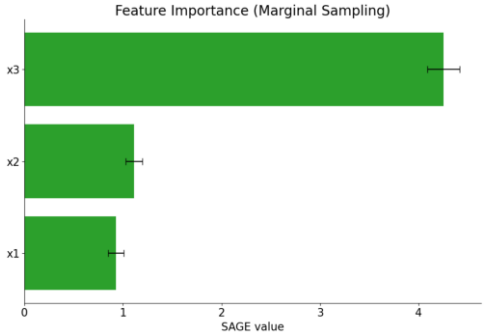
Simulations

True **linear** model: $Y = X_1 + X_2 + 2X_3$

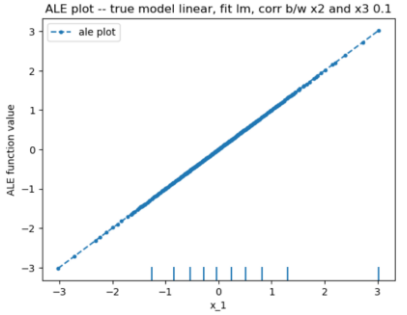
(X1, X2, X3) ~ Normal(0, *Cov*)

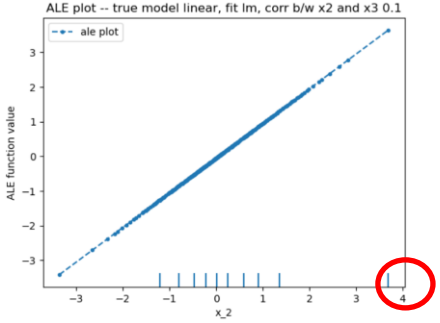
$$Cov = \begin{matrix} 1 & 0 & 0 \\ 0 & 1 & 0.1 \\ 0 & 0.1 & 1 \end{matrix}$$

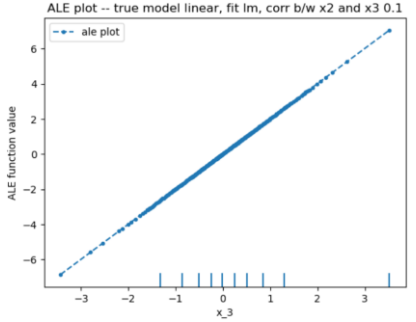
```
Permutation importance:
x3      1.264 +/- 0.040
x2      0.323 +/- 0.011
x1      0.308 +/- 0.010
```



ALE plots







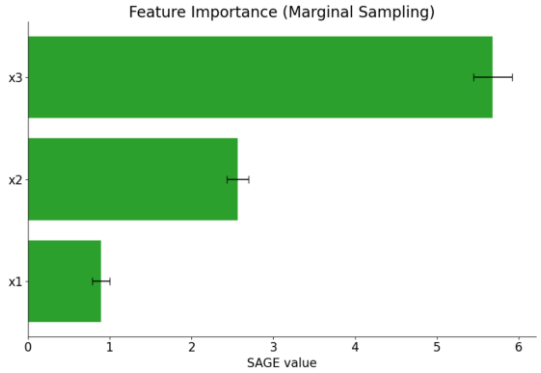
Simulations

True **linear** model: $Y = X_1 + X_2 + 2X_3$

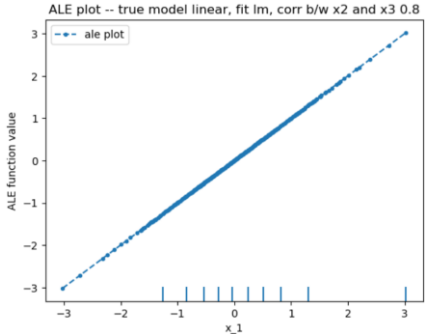
(X1, X2, X3) ~ Normal(0, *Cov*)

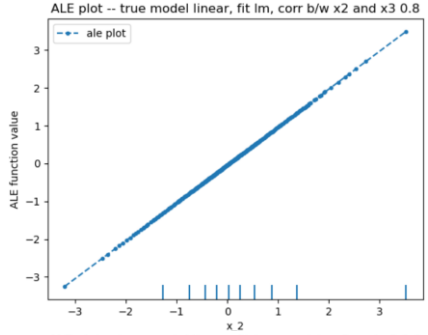
$$Cov = \begin{matrix} 1 & 0 & 0 \\ 0 & 1 & 0.8 \\ 0 & 0.8 & 1 \end{matrix}$$

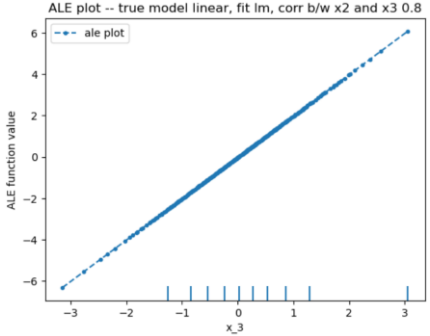
```
Permutation importance:
x3      0.884 +/- 0.031
x2      0.224 +/- 0.008
x1      0.214 +/- 0.007
```



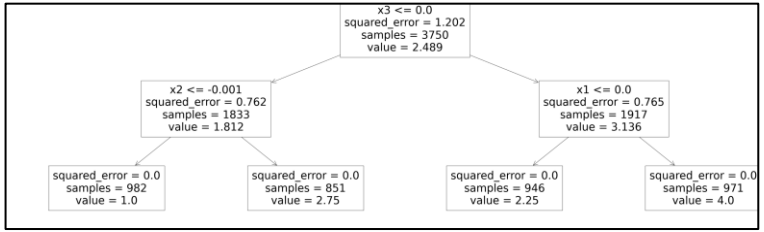
ALE plots







Simulations

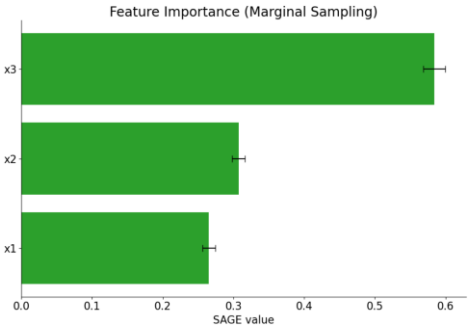


True **tree** model: $Y = \text{tree}(X_1, X_2, X_3)$

(X1, X2, X3) ~ Normal(0, *Cov*)

$$Cov = \begin{matrix} 1 & 0 & 0 \\ 0 & 1 & 0.1 \\ 0 & 0.1 & 1 \end{matrix}$$

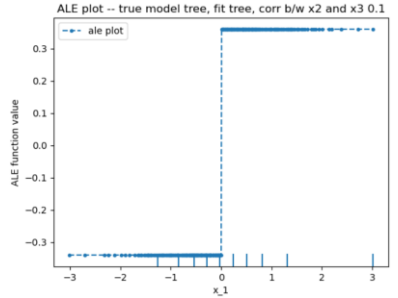
```
Permutation importance:
x3      1.344 +/- 0.056
x2      0.676 +/- 0.030
x1      0.646 +/- 0.026
```



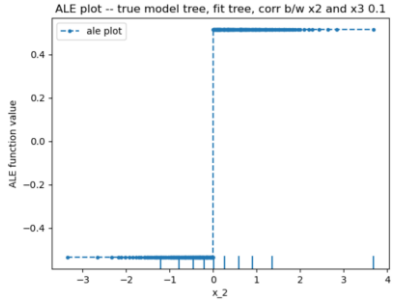

```
MDI(MSE): [0.32565713 0.30984958 0.36449329]
```

ALE plot does not agree that x3 is most important

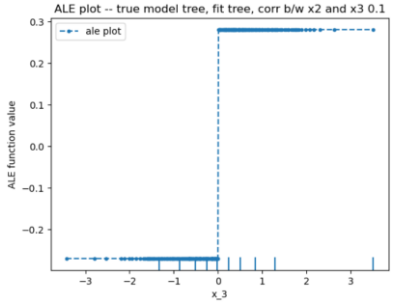
ALE plots



sd = 0.350

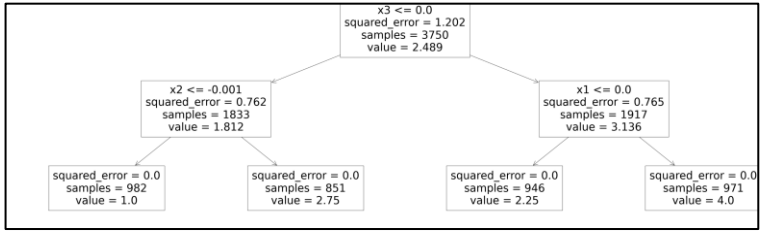


sd = 0.525



sd = 0.275

Simulations

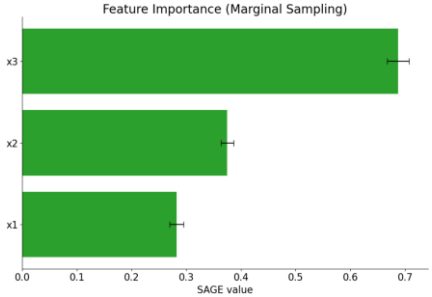


True **tree** model: $Y = \text{tree}(X_1, X_2, X_3)$

(X1, X2, X3) ~ Normal(0, *Cov*)

$$Cov = \begin{matrix} 1 & 0 & 0 \\ 0 & 1 & 0.8 \\ 0 & 0.8 & 1 \end{matrix}$$

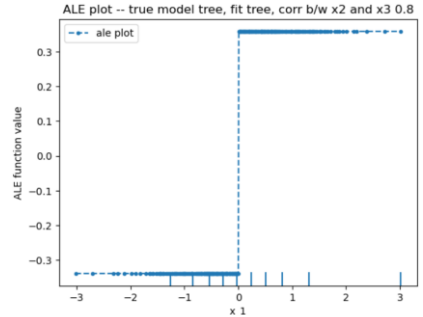
```
Permutation importance:
x3      1.152 +/- 0.044
x1      0.567 +/- 0.025
x2      0.549 +/- 0.019
```



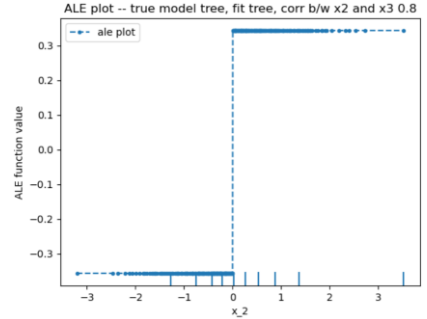

```
MDI(MSE): [0.27758803 0.16721372 0.55519825]
```

All agree that x3 is most important

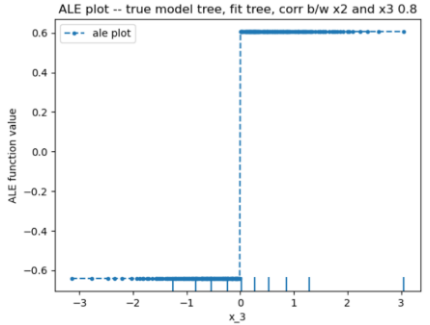
ALE plots



std = 0.348



std = 0.350



std = 0.624