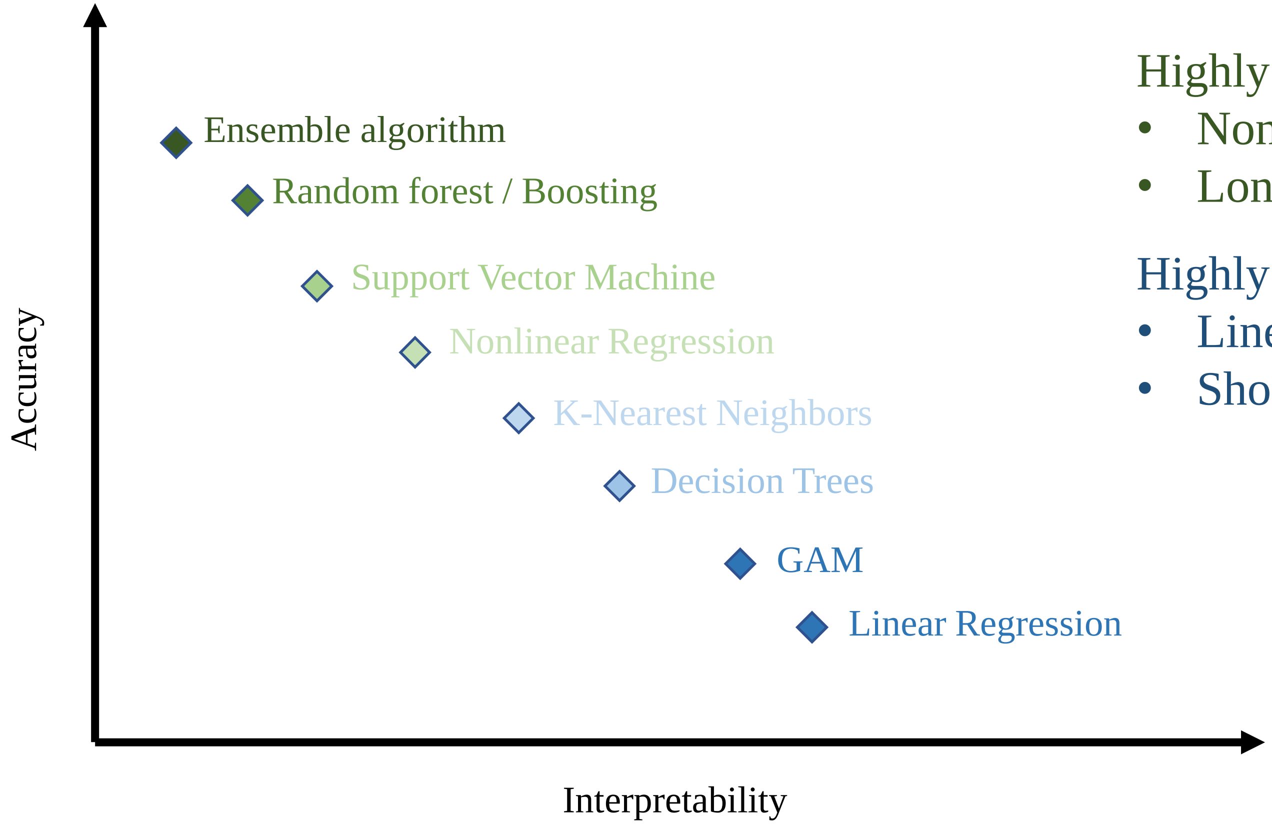
Interpretable Machine Learning (MSBA 7027)

Zhengli Wang

Faculty of Business and Economics
The University of Hong Kong
2023

Accurate vs Interpretable: a Tradeoff



Highly accurate models

- Non-linear & Non-smooth relationship
- Long computation time

Highly interpretable models

- Linear and smooth relationships
- Short computation time

Simple linear model: easily interpreted, but prediction not accurate for complex problems
Complex nonlinear model: better performance, but too complex for humans to understand

Achieve Interpretability: Two Options

Build interpretable ML models	Derive explanations for complex ML models
Model-based	**Post-hoc (Opposite of Ad-hoc)**

Categorization of Interpretable ML Methods: Overview

Agnosticity

- Model-agnostic: Applicable to all model types
- Model-specific: Only applicable to a specific model type

Scope

- Global scope: Explaining the whole model
- Local scope: Explaining individual predictions

Explanation type

- Visual
- Feature importance
- Surrogate

Categorization of Interpretable ML Methods: Agnosticity

Agnosticity

Model-agnostic: Applicable to all model types

e.g. Permutation-based feature importance, PDP, LIME, SHAP

Model-specific: Only applicable to a specific model type

e.g. Impurity-based feature importance (only applicable to tree-based method, e.g. RF, GBM)

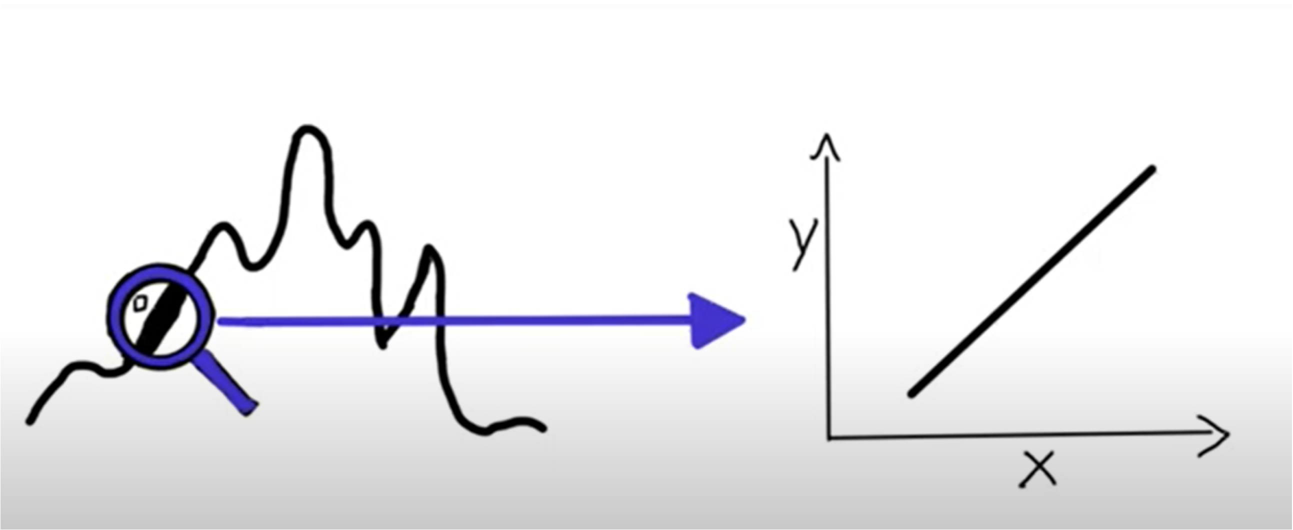
Categorization of Interpretable ML Methods: Agnosticity

Scope

Global scope: Explaining the whole model

Local scope: Explaining individual predictions

e.g. Linear regression coeff



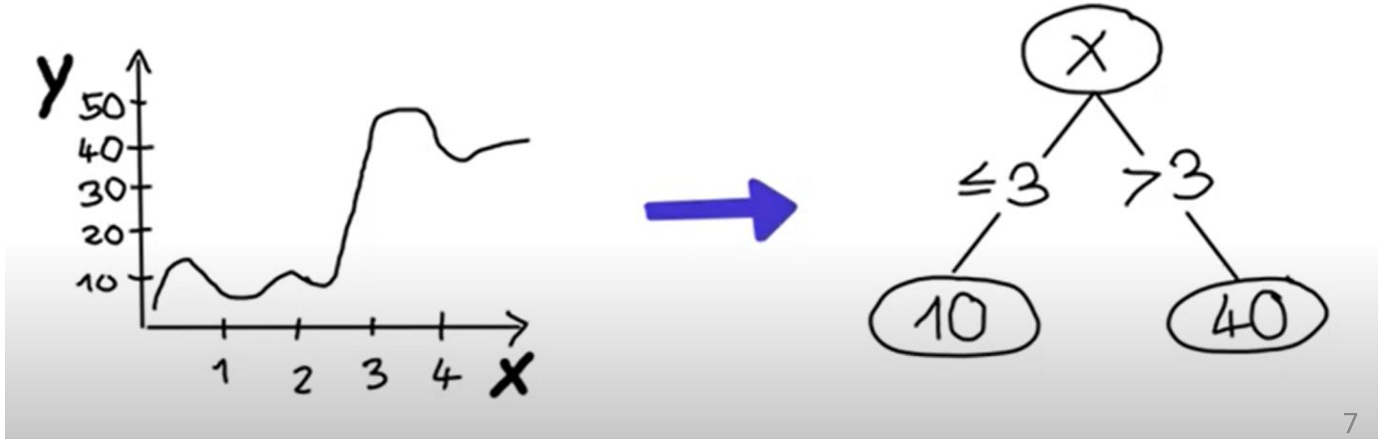
Categorization of Interpretable ML Methods: Agnosticity

Explanation type

- Visual
- Feature importance
- Surrogate

e.g. PDP

e.g. Impurity-based feature importance, Permutation-based feature importance



Achieve Interpretability: Two Options

Build interpretable ML models

Derive explanations for complex ML models

Model-based

Post-hoc
(Opposite of Ad-hoc)

Build Interpretable ML Models

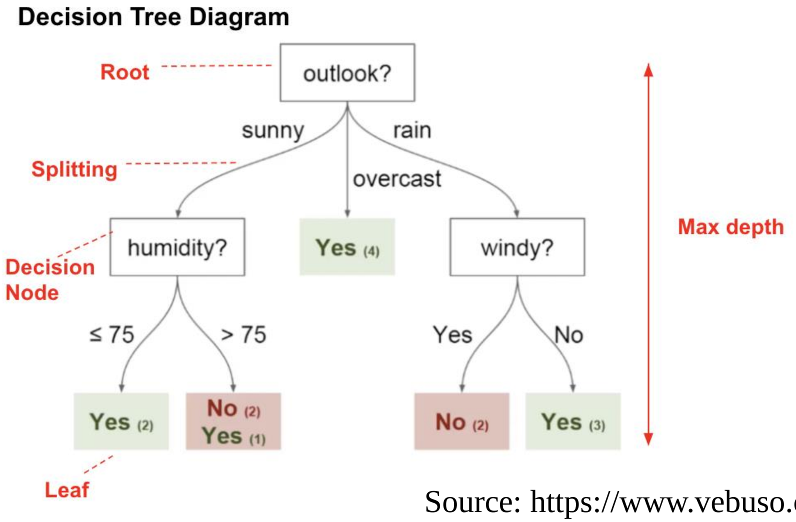
In many instances, simple models suffice, we do NOT always need complex models

- Linear regression
  - learns the coefficients (β) for a weighted sum of feature inputs

$$y = \beta_0 + \beta_1 x_1 + \cdots + \beta_n x_n$$

  - can directly interpret the impact of the inputs

- Decision Tree



Source: https://www.vebuso.com/2020/01/decision-tree-intuition-from-concept-to-application/

Achieve Interpretability: Two Options

Build interpretable ML models

Derive explanations for complex ML models

Model-Agnostic

Model-based

Post-hoc (Opposite of Ad-hoc)

Outline

- Permutation-based Feature Importance
- PDP
- LIME (Local Interpretable Model-Agnostic Explanations)
- SHAP (SHapley Additive exPlanations)

Permutation-based Feature Importance

Idea

If a feature is important, randomly permuting its value would make the resulting model worse

If a feature is NOT important, randomly permuting its values will likely keep the model error relatively unchanged

Intuition: suppose have two features x_1, x_2, true relationship: $y = 2x_1$

Permutation-based Feature Importance

Example

- Get a sample from the training data, fix a feature
- 1st row: benchmark error
    - Permute the feature value (to each values of the feature in the sample)
    - Compute the permuted error
    - Calculate: permuted error – benchmark error
- Repeat for every row in the sample

Permutation-based Feature Importance

Implementation: vip package

- vip (model_object, train = trainMatrix, method = "permute", target = y, metric = "RMSE", nsim = #simulations, sample_frac = frac, pred_wrapper = pred_wrapper)

- For general model_object
- trainMatrix needs to be a dataframe
- pred_wrapper: takes in [model_object & data] and outputs predictions

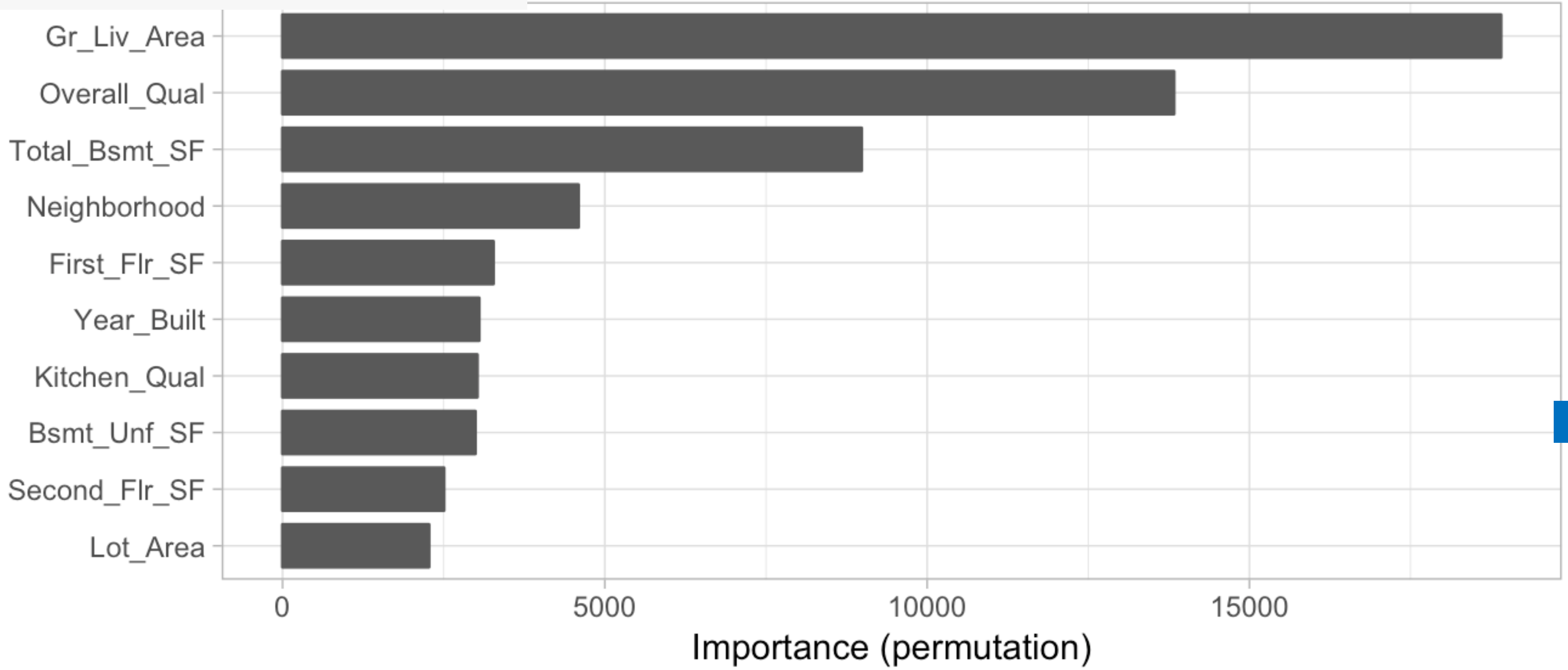
Permutation-based Feature Importance

```
pred <- function(object, newdata)  {
  results <- as.vector(h2o.predict(object, as.h2o(newdata)))
  return(results)
}
```

Takes in model object & newdata, and returns the predictions

```
vip(
  ensemble_tree,
  train = as.data.frame(train_h2o),
  method = "permute",
  target = "Sale_Price",
  metric = "RMSE",
  nsim = 5,
  sample_frac = 0.5,
  pred_wrapper = pred
)
```

Func from vip package



Sample 50% of training data
Repeat simulations 5 times

Permutation-based Feature Importance

Note

- Can become slow as #predictors ↑
- Can speed up computation by
  - ↓ sample size
  - ↓ #simulations
- However, these may make the feature importance estimates less accurate

Permutation-based Feature Importance

Summary

Expln. Type: Feature Importance

Scope: Global

Agnosticity: Model-Agnostic

PDP (Partial Dependence Plot)

Idea

Understand marginal effect of a feature on the predicted outcome

By marginal effect, we take into account the average effect of all the other features

PDP (Partial Dependence Plot)

Example

Feature of interest: Gr_Liv_Area

	Gr_Liv_Area	X1	X2	X3	…
1	687	0	a	2	…
2	334	0	c	6	…
3	2107	1	c	4	…
4	3329	0	b	2	…
5	5095	1	a	2	…

Construct a grid of j evenly spaced values across the range of Gr_Liv_Area
Say j = 20, the grid will consist of 20 values: 334, 585, 835, 1086, …, 5095

PDP (Partial Dependence Plot)

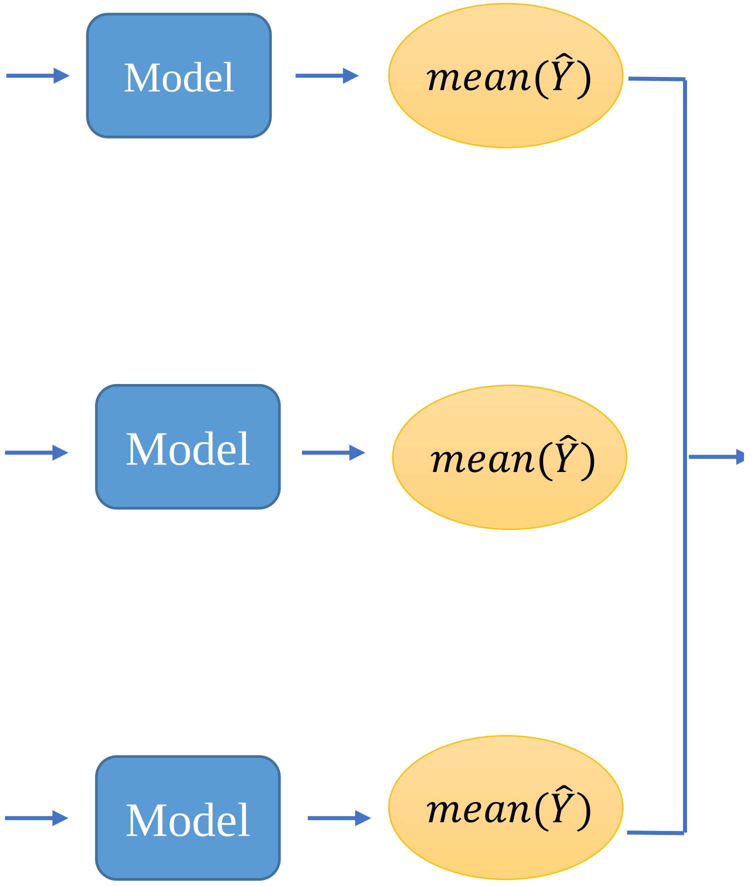
Example

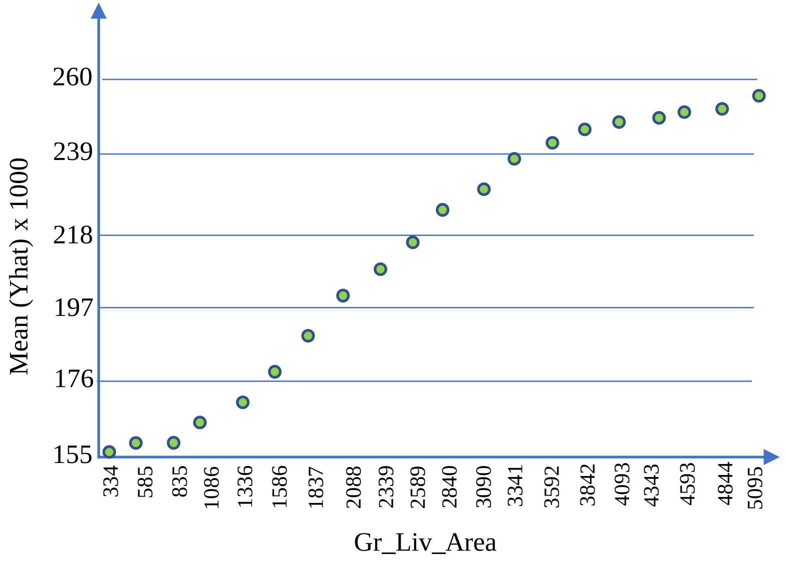
	Gr_Liv_Area	X1	X2	X3	…
1	334	0	a	2	…
2	334	0	c	6	…
3	334	1	c	4	…
4	334	0	b	2	…
5	334	1	a	2	…

	Gr_Liv_Area	X1	X2	X3	…
1	585	0	a	2	…
2	585	0	c	6	…
3	585	1	c	4	…
4	585	0	b	2	…
5	585	1	a	2	…

…

	Gr_Liv_Area	X1	X2	X3	…
1	5095	0	a	2	…
2	5095	0	c	6	…
3	5095	1	c	4	…
4	5095	0	b	2	…
5	5095	1	a	2	…





PDP (Partial Dependence Plot)

Implementation: pdp package

- Partial(model_object, train = trainMatrix, pred.var = “varName”, pred.fun = avg_pred_wrapper)

- For general model_object
- trainMatrix needs to be a dataframe
- avg_pred_wrapper: takes in [model_object & data] and outputs [the **average value** of the predictions]

PDP (Partial Dependence Plot)

Takes in model object & newdata, and returns the mean of predictions

```
# Custom prediction function wrapper
pdp_pred <- function(object, newdata)  {
  results <- mean(as.vector(h2o.predict(object, as.h2o(newdata))))
  return(results)
}
```

```
pdp_pred <- function(object, newdata)  {
  predObj = predict(object, newdata)
  #results <- mean(as.vector(predObj$predictions))
  results <- mean(as.vector(predObj))
  return(results)
}
```

```
# Compute partial dependence values
pd_values <- partial(
  ensemble_tree,
  train = as.data.frame(train_h2o),
  pred.var = "Gr_Liv_Area",
  pred.fun = pdp_pred,
  grid.resolution = 20
)
```

Func from pdp package

j = 20

PDP (Partial Dependence Plot)

Summary

Expln. Type: Visual

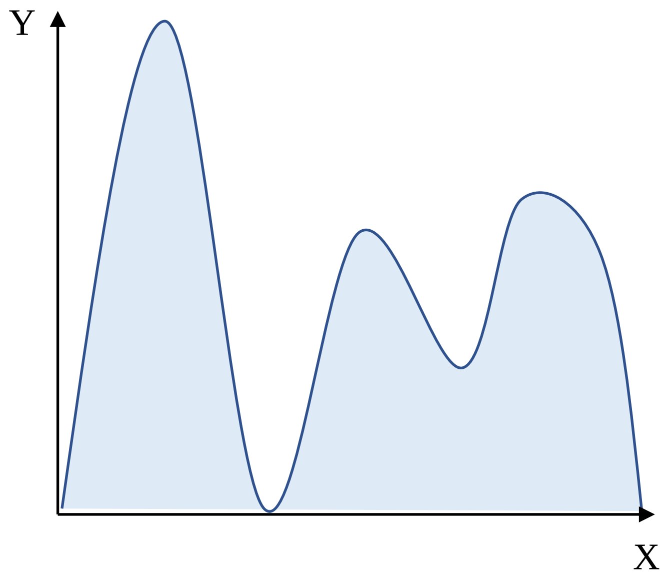
Scope: Global

Agnosticity: Model-Agnostic

LIME (Local Interpretable Model-Agnostic Explanations)

Expln. Type Surrogate Models

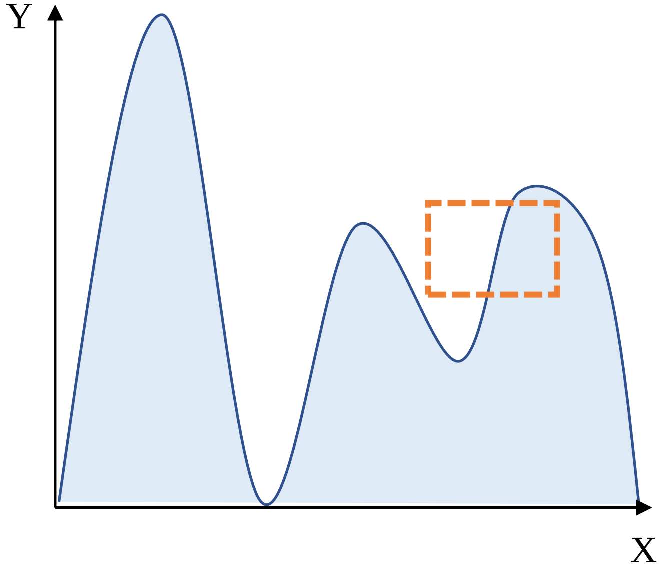
Idea Locally, every complex model can be approximated by a simple model



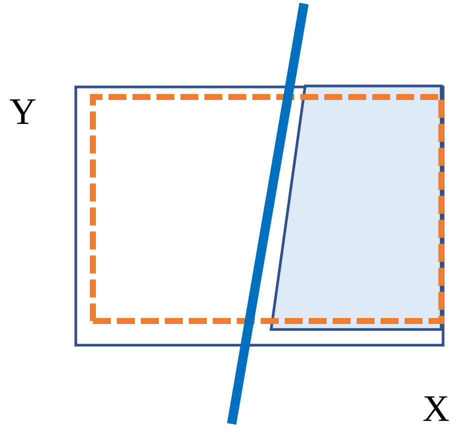
LIME (Local Interpretable Model-Agnostic Explanations)

Expln. Type Surrogate Models

Idea Locally, every complex model can be approximated by a simple model



Local linear model



LIME (Local Interpretable Model-Agnostic Explanations)

$$\xi(x) = \underset{g \epsilon G}{\operatorname{argmin}} \mathcal{L}(f, g, \pi_x) + \Omega(g)$$

LIME (Local Interpretable Model-Agnostic Explanations)

$$\xi(x) = \underset{g \epsilon G}{\operatorname{argmin}}\, \overbrace{\mathcal{L}(f, g, \pi_x)}^{\text{Loss Func (between f and g)}} + \overbrace{\Omega(g)}^{\text{Penalty}}$$

New data

Family of interpretable models

Complex model

Simple interpretable model

Weight func (Proximity func)

LIME (Local Interpretable Model-Agnostic Explanations)

$$\xi(x) = \underset{g \epsilon G}{\operatorname{argmin}} \mathcal{L}(f, g, \pi_x) + \Omega(g)$$

How?

1. Generate new datapoints (get distribution from training data)

2. Get predictions from the complex model f

This results in a new dataset Z with
- Labels: Prediction of complex model
- Features: Newly generated datapoints

(This slide is intended to be empty)

LIME (Local Interpretable Model-Agnostic Explanations)

$$\xi(x) = \underset{g \epsilon G}{\operatorname{argmin}}\, \mathcal{L}(f, g, \pi_x) + \Omega(g)$$

e.g. $(f, g, \pi_x) = \sum_{z \in \mathrm{Z}} \pi_x(\mathrm{z})\big(f(z) - g(z)\big)^2$

Penalty func
e.g. LASSO

LIME (Local Interpretable Model-Agnostic Explanations)

Summary

Expln. Type: Surrogate Models

Scope: Local

Agnosticity: Model-Agnostic

SHAP (SHapley Additive exPlanations)

Motivation

- Limitation of observing single feature effect at a time (e.g. Perm-based FIM, PDP)
  - Miss **interaction** between features
  - May produce misleading explanations for the ML model

- Solution: Borrows idea from **cooperative game theory**

SHAP (SHapley Additive exPlanations)

Expln. Type Feature Importance

Idea Cooperative Game Theory

- Imagine several cooperative players in a game
- After the game is over, receive certain payoff
- **Problem:** Divide the payoff among players, in a **fair** way

- **Answer:** Shapley values, which describes the average contribution of each player

SHAP (SHapley Additive exPlanations) – Game Theory

Fairness is tricky

- E.g. each individual tends to think he/she contributes the most

Let v be the payoff function, v: a set of player $\rightarrow$ 1-dim number (reward)

SHAP (SHapley Additive exPlanations) – Game Theory

Fairness is tricky

- E.g. each individual tends to think he/she contributes the most

Let v be the payoff function, v: a set of player → 1-dim number (reward)

First idea: pay everyone by his/her marginal contribution

- This idea does NOT work in the following scenario

$$p = 4 \text{ players}, v(\{1,2,3,4\}) = 10000, v(S) = 0 \text{ for all } S \neq \{1,2,3,4\}$$

Intuitively, how much should we pay each individual?

SHAP (SHapley Additive exPlanations) – Game Theory

Suppose there are p players: player 1,2,3,…,p

Shapley value of individual j

$$\phi_j = \sum_{S \subseteq \{1,\ldots,p\} \backslash \{j\}} \frac{|S|!\,(p - |S| - 1)!}{p!} [v(S \cup \{j\}) - v(S)]$$

SHAP (SHapley Additive exPlanations) – Game Theory

Example cont'd

p = 4 players, $v(\{1,2,3,4\}) = 10000$, $v(S) = 0$ for all $S \neq \{1,2,3,4\}$

SHAP (SHapley Additive exPlanations) – Game Theory

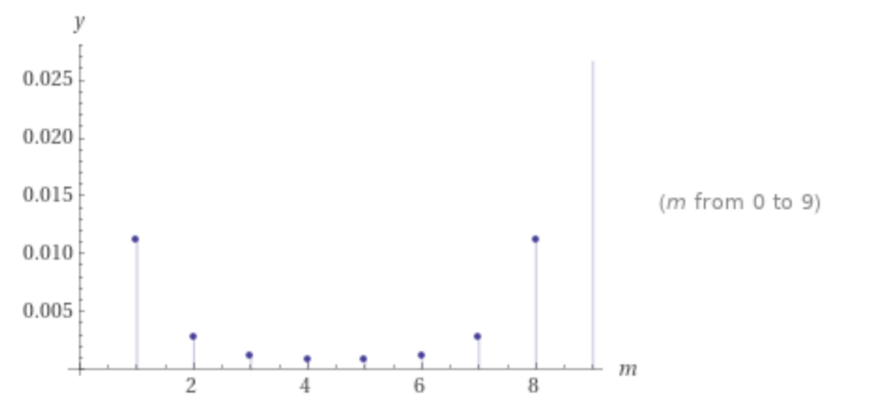
Another Example

A	B	C
$60 Coupon	$40 Coupon	$30 Coupon
Small	Medium	Large
500	750	1000
$70 Coupon	$90 Coupon	$110 Coupon

SHAP (SHapley Additive exPlanations) – Game Theory

Intuition of Weighting: A player’s contribution should be weighed more if
1. The game already has lots of players; 2. The game has only a few players

plot $m! \times \frac{(10 - m - 1)!}{10!}$



SHAP (SHapley Additive exPlanations)

From Game Theory to Machine Learning

- Think of features as players, prediction outcome from model as payoff

SHAP (SHapley Additive exPlanations)

Shapley value of feature *j*

Complex model

Input datapoint

Shapley value for feature j

$$\phi_j(f,x) = \sum_{S \subseteq \{1,\ldots,p\} \setminus \{j\}} \underbrace{\frac{|S|!\,(p-|S|-1)!}{p!}}_{\text{Weighting}} \underbrace{[f_x(S \cup \{j\}) - f_x(S)]}_{\text{Marginal Contribution of feature j}}$$

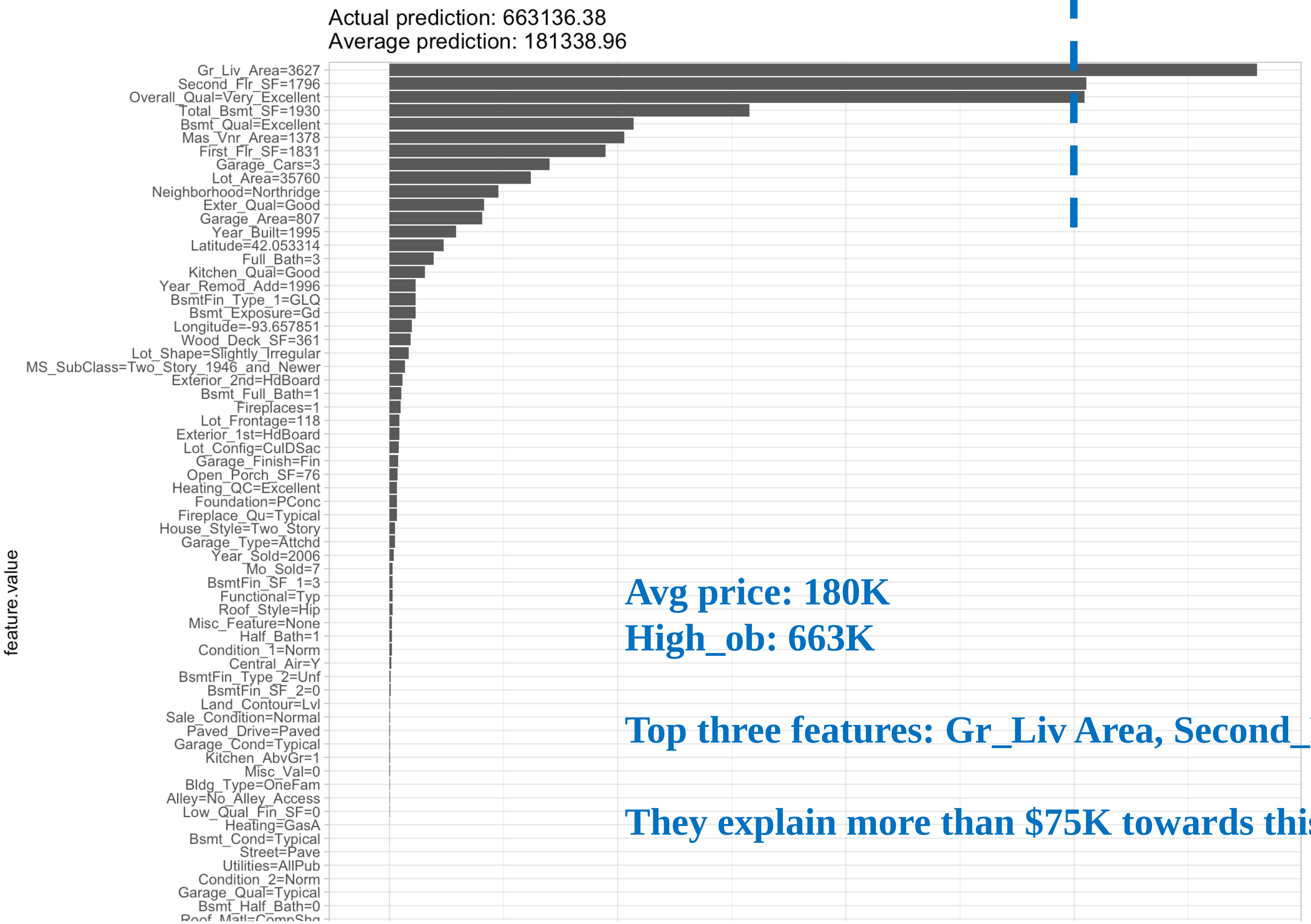
Subset of features excluding j

p : total #features

SH

Sha

75K



Avg price: 180K

High_ob: 663K

Top three features: Gr_Liv Area, Second_Flr_SF, Over_Qual

They explain more than $75K towards this price difference

SHAP (SHapley Additive exPlanations)

Summary

Expln. Type: Feature Importance

Scope: Local

Agnosticity: Model-Agnostic

End