Explainable AI - Introduction

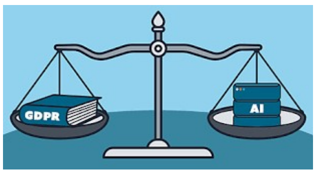
Lecture at

“**48th Winter Conference in Statistics**”

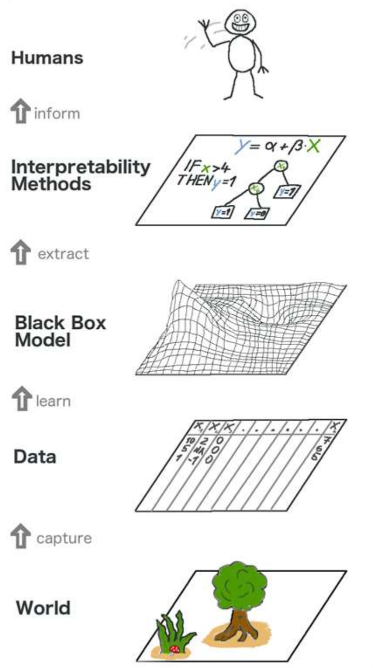
March 11th, 2024

Black box methods

- In many applications, machine-learning (ML) methods like deep neural networks, random forests and gradient boosting machines are currently outperforming more traditional statistical methods.
- Often it is hard to understand why these methods perform so well – there is usually a tradeoff between complexity and interpretability.



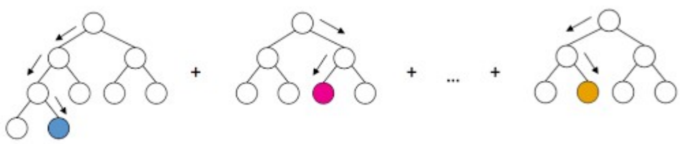
2



From the book
“Interpretable Machine Learning”.

Example: Mortgage robot

- XGBoost model which predicts mortgage default
- 28 covariates extracted from 6 transaction time series
  - Example 1: Mean value of the checking account during the last 365 days
  - Example 2: Standard deviation of the savings account during the last 365 days
- Why was Ola Nordmann rejected a loan?



4

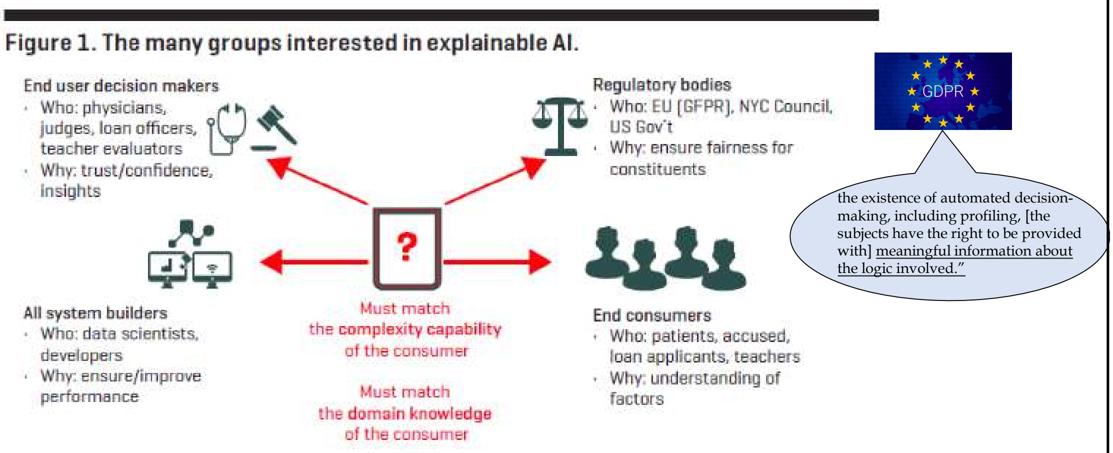


Figure 1. The many groups interested in explainable AI (from Hind, 2019)

Difficult problem

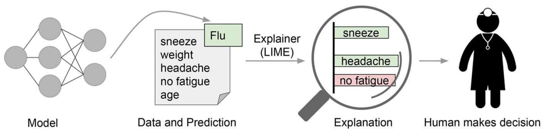
- Difficult problem!
  - Not even for the simple linear regression model it is straightforward to determine the importance of each variable if the variables are not independent!

$$y = 0.5\,x_1 + 0.2\,x_2 + 0.3\,x_3 + \epsilon$$

What is the global importance of x_1, x_2 and x_3?

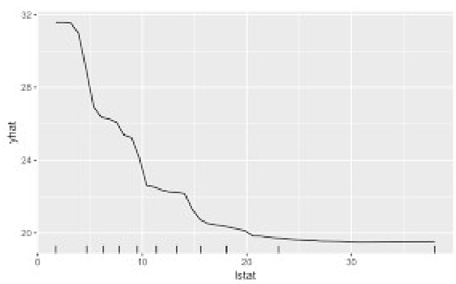
Global and local explanations

- **Global explanations:** Overall significance of variables
- **Local explanations:** Aims to interpret individual predictions



Global explanation methods

- Model-specific:
  - Standardized coefficients (linear model)
  - Shapley regression (linear model)
  - Gini importance (Random forest/XGBoost)
- Model-agnostic
  - Partial dependence plots (PDP)
  - Accumulated Local Effects (ALE) plots



Local explanation methods

- Model-specific methods:
  - Deep Lift: For deep learning models
  - TreeSHAP: For XGBoost models
- Model-agnostic methods:
  - LIME Local linear regression
  - Shapley Based on concepts from game theory
  - Counterfactual explanations: Which variables should be altered to obtain a different decision?

9

Global importance measures

Linear model

$$y = 0.5\,x_1 + 0.2\,x_2 + 0.3\,x_3 + \epsilon$$

What is the global importance of x_1, x_2 and x_3?

11

Answer

- The answer is different depending on whether the three variables have equal variances and whether they are independent or not.
  - If they are independent and have equal variances the regression coefficients may be used to assess how important they are.
  - If they have different variances, one must use standardised coefficients to assess their importance.
  - If they are not independent, more advanced methods must be used, where one takes into account both a variable's contribution by itself and in contribution with other predictor variables.

Standardised coefficients

- The importance of a feature in a linear regression model can be measured by the absolute value of its test t-statistic:

$$t_{\hat{\beta}_j} = \frac{\hat{\beta}_j}{SE(\hat{\beta}_j)}$$

- Not a good measure if the features are not independent.
- A feature with high positive correlation with the response *y* and another feature might get a negativ coefficient, because, *given the other feature*, it is negatively correlated with *y*.

Averaging over orderings method (lmg)

- As long as regressors are independent, assessment of relative importance in a linear regression model is simple.
- However, in real life regressors are typically dependent.
- The averaging over orderings method (lmg) proposed by Lindeman, Merenda and Gold (1980) decomposes R^2 into non-negative contributions that sum to the total R^2 in a way that takes the dependence between the regressors into account.

$$R^2 = \frac{\text{Model SS}}{\text{Total SS}} = \frac{\sum_{i=1}^{n}(\hat{y}_i - \bar{y})^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}$$

lmg

- Let $r = (r_1, \dots, r_p)$ be a random order of the regressors $x_1, \dots, x_p$ and let $S_k(r)$ be the set of regressors entered into the model before x_k in order r.
- Then, the portion of R^2 allocated to x_k in order r is

$$seqR^2(\{x_k\}|S_k(r)) = R^2(\{x_k\} \cup S_k(r)) - R^2(S_k(r))$$

- and lmg for variable x_k is defined as

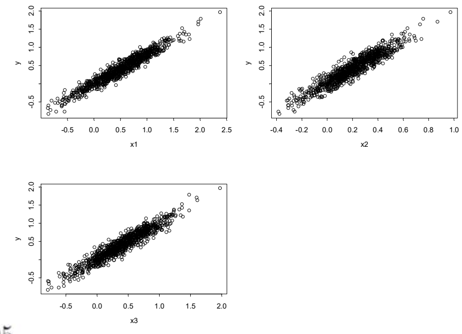
$$\mathrm{LMG}(x_k) = \frac{1}{p!} \sum_{r \text{ permutation}} seqR^2(\{x_k\}|r).$$

Example

- Three variables x_1, x_2, x_3
- 6 different permutations: 123 213 312 132 231 321
- $\mathrm{LMG}(x_1) = \frac{1}{6}(R^2(x_1) - R^2(0) + R^2(x_1, x_2) - R^2(x_2) + R^2(x_1, x_3) - R^2(x_3) + R^2(x_1) - R^2(0) + R^2(x_1, x_2, x_3) - R^2(x_2, x_3) + R^2(x_1, x_2, x_3) - R^2(x_2, x_3))$
  $= \frac{1}{3}(R^2(x_1) - R^2(0)) + \frac{1}{6}(R^2(x_1, x_2) - R^2(x_2)) + \frac{1}{6}(R^2(x_1, x_3) - R^2(x_3)) + \frac{1}{3}(R^2(x_1, x_2, x_3) - R^2(x_2, x_3))$
- As will be shown later, the formula in red is exactly the same formula that is used in **Shapley regression!**

Example

$$y = 0.5\,x_1 + 0.2\,x_2 + 0.3\,x_3 + \epsilon$$
$$x_1 \sim N(0.5, 0.5^2)$$
$$x_2 \sim N(0.2, 0.2^2)$$
$$x_3 \sim N(0.4, 0.4^2)$$
$$\epsilon \sim N(0, 0.1^2)$$
$$\text{cor}(x_1, x_2) = \text{cor}(x_1, x_3) = \text{cor}(x_2, x_3) = 0.95$$



What is the global importance of x_1, x_2 and x_3?

Global variable importance

	X1	X2	X3
Regression coef.	0.50	0.20	0.30
Standardized coef.	0.59	0.12	0.29
lmg	**0.35**	**0.32**	**0.33**

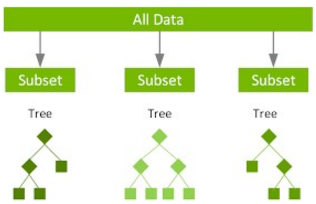
The three covariates are almost equally important!

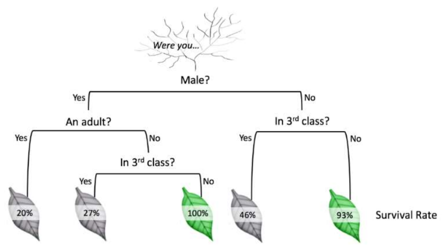
Software

- The R-package relaimpo provides several metrics for assessing relative importance in linear models:
- https://cran.r-project.org/web/packages/relaimpo/index.html

XGBoost

- XGBoost (eXtreme Gradient Boosting) is an effective implementation of gradient boosting with trees as basis models.
- Boosting is an ensemble technique where new models are added to correct the errors made by existing models. Models are added sequentially until no further improvements can be made.
- State-of-art for tabular data.

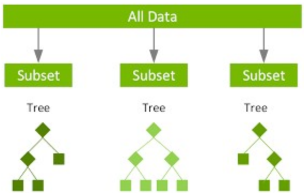




Tree for Titanic survival prediction

Variable importance XGBoost

- Importance is calculated for a single decision tree by the amount that each feature split point improves the performance measure, weighted by the number of observations the node is responsible for.
- The performance measure may be the gini impurity or entropy used to select the split points or another more specific error function.
- The feature importances are then averaged across all of the decision trees within the model.



Permutation-based feature importance

- Estimate the original model error $e^{orig} = L(y, f(X))$ (e.g. mean squared error)
- For each feature $k = 1,\ldots,p$ do:
  - Generate the feature matrix X^{perm} by permuting feature k in the data X. This breaks the association between feature k and true outcome y.
  - Estimate error $e^{perm} = L(Y,f(X^{perm}))$ based on the predictions of the permuted data.
  - Calculate permutation feature importance $FI(k)= e^{perm}/e^{orig}$. Alternatively, the difference can be used: $FI(k) = e^{perm} - e^{orig}$
- Sort features by descending FI.

Software

- PYTHON:
  - See e.g.: https://machinelearningmastery.com/feature-importance-and-feature-selection-with-xgboost-in-python/
- R:
  - XGBoost: https://cran.r-project.org/web/packages/xgboost/index.html
  - RandomForest: https://cran.r-project.org/web/packages/randomForest/randomForest.pdf

Bike rental data set

- This dataset contains daily counts of rented bicycles from the bicycle rental company Capital-Bikeshare in Washington D.C., along with weather and seasonal information. The data was kindly made openly available by Capital-Bikeshare. Fanaee-T and Gama (2013)[13] added weather data and season information. The goal is to predict how many bikes will be rented depending on the weather and the day. The data can be downloaded from the UCI Machine Learning Repository or as an .Rdata file from https://github.com/christophM/ interpretable-ml-book/blob/master /data/bike.Rdata

- For Python users the following link may be useful:

https://www.storybench.org/tidytuesday-bike-rentals -part-2-modeling-with-gradient-boosting-machine/

Source:clipart-library.com

Bike rental data set

We use the slightly modified version that can be downloaded at https://github.com/christophM/interpretable-ml-book/blob/master/data/bike.Rdata

Variables:

- cnt: Count of bicycles including both casual and registered users (response).
- season: Spring, summer, fall or winter.
- holiday: Indicator whether the day was a holiday or not.
- yr: Either 2011 or 2012.
- days_since_2011: Number of days since the 01.01.2011 (the first day in the dataset).
- working_day: Indicator whether the day was a working day or weekend.
- weekday: Day of week ('SUN', 'MON', 'TUE', 'WED', 'THU', 'FRI', 'SAT')
- weatersit: The weather situation on that day. One of:
  - clear, few clouds, partly cloudy, cloudy
  - mist + clouds, mist + broken clouds, mist + few clouds, mist
  - light snow, light rain + thunderstorm + scattered clouds, light rain + scattered clouds
  - heavy rain + ice pallets + thunderstorm + mist, snow + mist
- temp: Temperature in degrees Celsius.
- hum: Relative humidity in percent (0 to 100).
- wind_speed: Wind speed in km per hour.

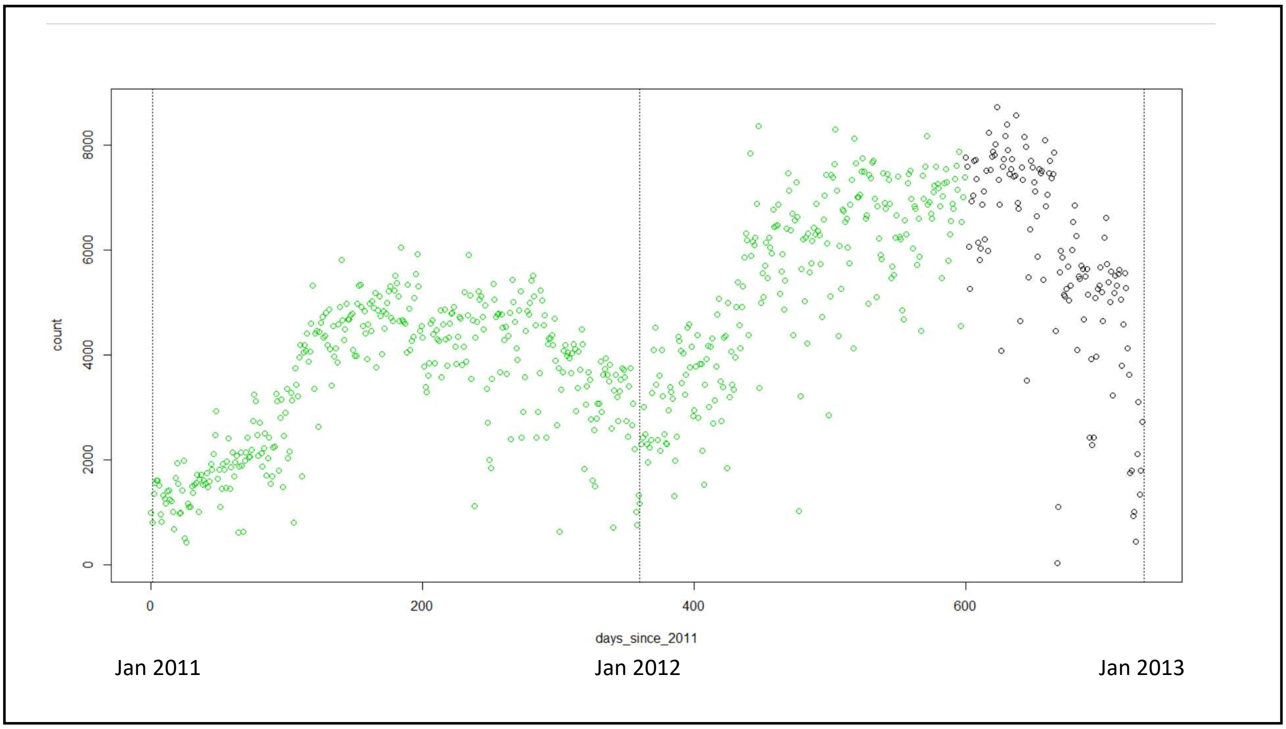
Linear model

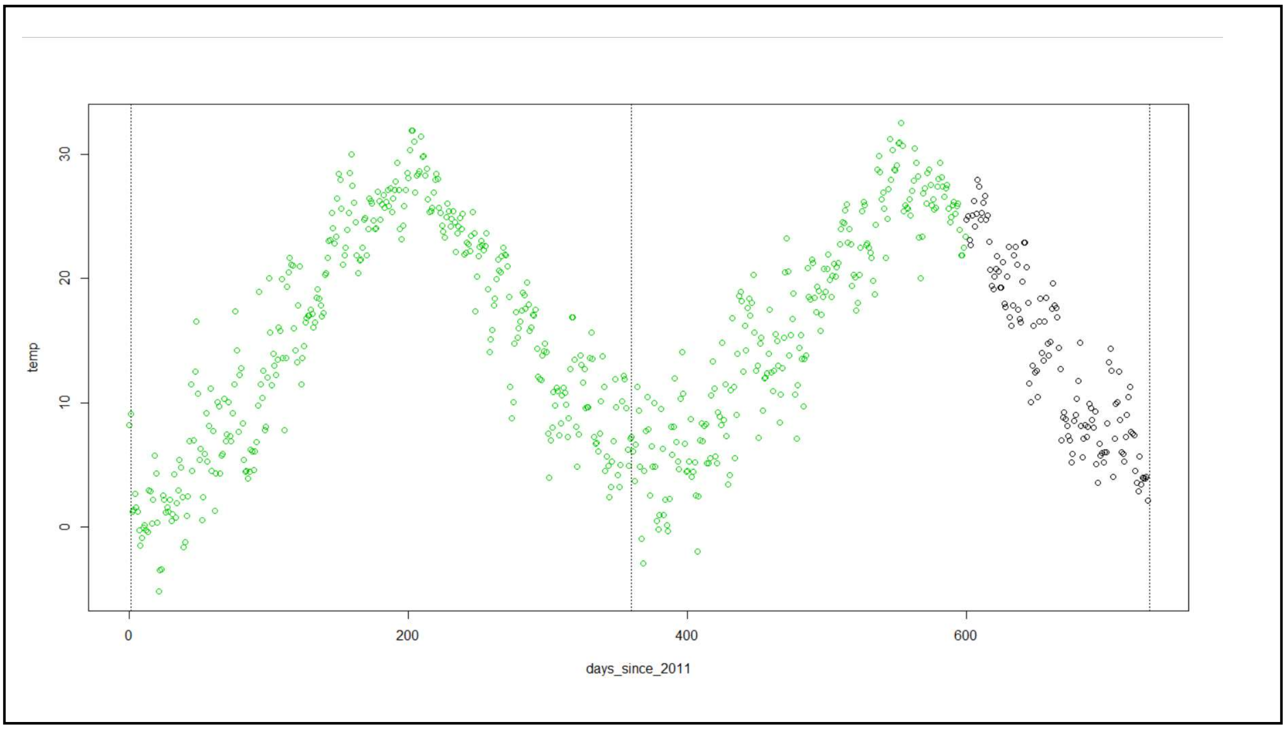
	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	2190.601764	227.898062	9.6122001	2.242291e-20
weathersitRAIN/SNOW/STORM	-1752.752609	193.040301	-9.0797238	1.761882e-18
temp	79.196992	8.725721	9.0762695	1.811416e-18
windspeed	-39.983442	6.226363	-6.4216365	2.839916e-10
weathersitMISTY	-403.354102	77.755824	-5.1874456	2.965845e-07
seasonWINTER	1022.757860	219.109652	4.6677901	3.797994e-06
hum	-12.557802	2.840354	-4.4212094	1.175172e-05
seasonSUMMER	714.940108	171.970442	4.1573430	3.714002e-05
mnthMAR	805.788971	239.819971	3.3599744	8.315280e-04
weekdaySAT	325.113525	106.550391	3.0512654	2.384641e-03
mnthMAY	1309.462838	444.838414	2.9436820	3.375219e-03
weekdayFRI	302.936901	107.386611	2.8209932	4.954018e-03
weekdayTHU	294.438172	107.411555	2.7412151	6.312990e-03
mnthAPR	919.475736	356.583160	2.5785731	1.017017e-02
weekdayTUE	272.030492	106.815486	2.5467327	1.113497e-02
seasonFALL	561.596710	223.565166	2.5120045	1.227945e-02
mnthJUN	1265.562753	535.082099	2.3651749	1.835486e-02
weekdayWED	239.718882	107.264136	2.2348465	2.581343e-02
holidayHOLIDAY	-401.693918	187.024915	-2.1478097	3.214902e-02
yr2012	2525.594555	1226.983365	2.0583772	4.000713e-02
mnthSEP	1616.876600	816.012604	1.9814358	4.802162e-02
mnthAUG	1265.356448	715.832090	1.7676721	7.764975e-02
mnthOKT	1470.378330	934.992999	1.5726089	1.163633e-01
mnthJUL	929.201306	632.007193	1.4702385	1.420480e-01
mnthFEB	234.179270	162.412800	1.4418769	1.498853e-01
weekdayMON	138.397329	109.498064	1.2639249	2.067727e-01
mnthNOV	1131.008491	1034.979687	1.0927833	2.749498e-01
mnthDEZ	1104.438624	1128.543616	0.9786406	3.281720e-01
days_since_2011	-1.257864	3.357210	-0.3746754	7.080410e-01

```
linearMod <- lm(cnt~.,data=bikeTrain)
tmp <- summary(linearMod)
tmp$r.square
tmp$coefficients[rev(order(abs(tmp$coefficients[,3]))),]
```

	cnt	temp	hum	wind	days
cnt	**1.00**	**0.70**	**-0.12**	**-0.21**	**0.70**
temp	**0.70**	1.00	0.12	-0.17	0.33
hum	**-0.12**	0.12	1.00	-0.26	-0.04
windspeed	**-0.21**	-0.17	-0.26	1.00	-0.11
days_since_2011	0.70	0.33	0.04	-0.11	1.00

Why isn't "days_since_2011" regarded to be more important and why is the regression coeff. negative?


count
8000
6000
4000
2000
0
0
200
400
600
days_since_2011
Jan 2011
Jan 2012
Jan 2013


temp
30
20
10
0
0
200
400
600
days_since_2011

lmg-method

- Clear dependence between days_since_2011 and temp!
- Use the lmg-method to assess variable importance.

```
#Shapley regression
library("relaimpo")
crf <- calc.relimp(cnt~.,data=bikeTrain[,-6],type ="lmg", rela = TRUE )
rev(sort(crf$lmg))
```

- Variable importance weights:

days_since_2011	temp	mnth	yr	season	weathersit	windspeed	hum	weekday	holiday
0.218726846	0.186903959	0.180877090	0.167059928	0.141628581	0.058487497	0.023041686	0.016835282	0.004340669	0.002098463

- With this method, days_since_2011 has positive sign and is regarded to be the most important variable…

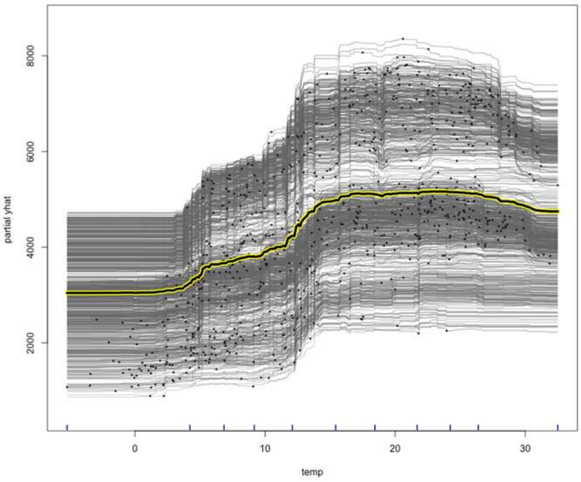
XGBoost

	Feature	Gain	Cover	Frequency
1:	days_since_2011	0.7976586942	0.3430909388	0.162966462
2:	temp	0.1132748771	0.2445233763	0.239017478
3:	hum	0.0295591111	0.1416631672	0.180444025
4:	weathersit	0.0213787474	0.0364816629	0.049598488
5:	windspeed	0.0192767348	0.0998856850	0.145488899
6:	weekday	0.0055235470	0.0546554218	0.091639112
7:	mnth	0.0054333041	0.0396486562	0.056683987
8:	season	0.0031174028	0.0115831467	0.024090694
9:	workingday	0.0023060841	0.0175030328	0.033065659
10:	holiday	0.0021865526	0.0105916387	0.013698630
11:	yr	0.0002849447	0.0003732736	0.003306566

```
library(xgboost)
n <- dim(bike)[1]
bikeTrain <- bike[1:600,]
bikeTest  <- bike[601:n,]

xgb.train <- xgb.DMatrix(data = as.matrix(sapply(bikeTrain[,-11], as.numeric)),label = bikeTrain[,"cnt"])
xgb.test  <- xgb.DMatrix(data = as.matrix(sapply(bikeTest[,-11], as.numeric)),label = bikeTest[,"cnt"])

params <- list(eta =  0.1,
          objective = "reg:squarederror",
          eval_metric = "rmse",
          tree_method="hist") # gpu_hist

RNGversion(vstr = "3.5.0")
set.seed(12345)
model <- xgb.train(data = xgb.train,
            params = params,
            nrounds = 50,
            print_every_n = 10,
            ntread = 5,
            watchlist = list(train = xgb.train,
                        test = xgb.test),
            verbose = 1)

xgb.importance(model=model)
```

ICE plot

- Individual Conditional Expectation (ICE) plots, plots one curve for each variable of interest and each observation.
- This is done by changing the variable of interest to be equal to the possible values in the data set.



Have fitted a Random Forest to the bike data set. Each grey line is the temp ICE curve for one specific observation.

Partial dependence (PD) plot

- The PD plot shows the marginal effect one feature has on the predicted outcome of the machine learning (ML-)model.
- Let x_S be the feature for which the PD should be plotted and x_C the other features used in the ML-model. Then, the PD is defined as

$$\hat{f}_{x_S}(x_S) = E_{x_C}\left[\hat{f}(x_S, x_C)\right] = \int \hat{f}(x_S, x_C)\, g(x_C) dx_C$$

- which can be approximated by

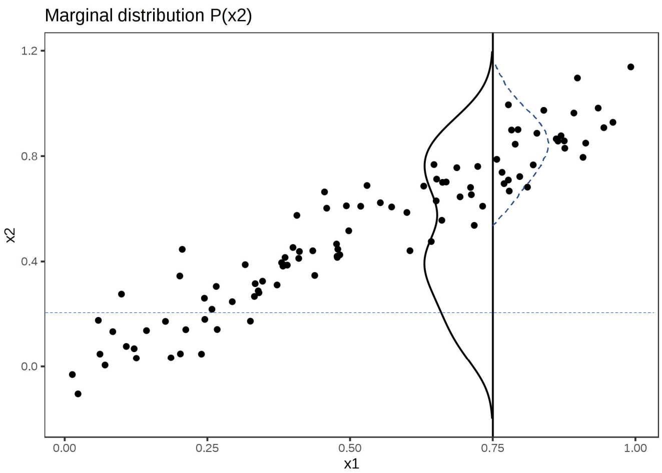
$$\hat{f}_{x_S}(x_S) = \frac{1}{n}\sum_{i=1}^{n} \hat{f}(x_S, x_C^{(i)})$$

Here, $x_C^{(i)}$; $i = 1, .., n$ are the values of the training observations for all features except x_S .

Partial dependence plot

- PD plots involve a serious pitfall if the predictor variables are far from independent, which is quite common with large observational data sets.
- In such cases, PD plots might require extrapolation of the response at predictor values that are very unlikely or even impossible.

Example



To calculate the feature effect of x1 at 0.75, the PD plot replaces x1 of all instances with 0.75, falsely assuming that the distribution of x2 at x1 = 0.75 is the same as the marginal distribution of x2 (vertical line). This results in unlikely combinations of x1 and x2 (e.g. x2=0.2 at x1=0.75), which the PD plot uses for the calculation of the average effect.

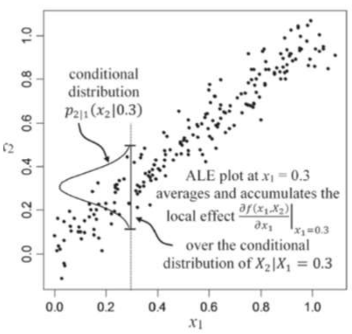
The figure is from https://christophm.github.io/interpretable-ml-book/ale.html

Software

- PYTHON:
  - PD plots are built into scikit-learn and you can use PDPBox
- R:
  - pdp R package: https://cran.r-project.org/web/packages/pdp/index.html
  - iml R package: https://cran.r-project.org/web/packages/iml/index.html

Accumulated local effects (ALE) plot

- Accumulated local effects plots (Apley and Zhu, 2006) is an alternative way of visualizing variable effects which do not require unreliable extrapolation with correlated variables.



Source: Apley and Zhu (2006)

Assume that we have only two variables. The main effect of X_1 is

$$f_{1,ALE}(x_1) = \int_{x_{min,1}}^{x1} \int p_{2|1}(x_2|z_1)\frac{\delta f(z_1,x_2)}{\delta z_1}dx_2\,dz_1 - \text{constant}$$

1) $\frac{\delta f(z_1,x_2)}{\delta z_1}$ is the local effect of z_1 on $f\,(z_1, x_2)$ at (z_1, x_2)

2) Average this effect over all values of x_2 with weights p($x_2\,|\,z_1$)

3) Accumulate the averaged effect over all values of z_1 in the interval $[x_{min,1}, x_1]$.

$x_{min,1}$ is a value just below min{$x_{i,1}$;i=1,..,n}.

Accumulated local effects (ALE) plot

- In the general multivariate case, ALE for feature *j* is defined as

$$f_{j,ALE}(x_j) = \int_{x_{min,j}}^{x_j} \int p(\boldsymbol{x}_{\setminus j}|z_j) \frac{\delta f(z_j, \boldsymbol{x}_{\setminus j})}{\delta z_j} \, d\boldsymbol{x}_{\setminus j} \, dz_j - \text{constant}$$

$$= g_{j,ALE}(x_j) - \int p(z_j) g_{j,ALE}(z_j) \, dz_j$$

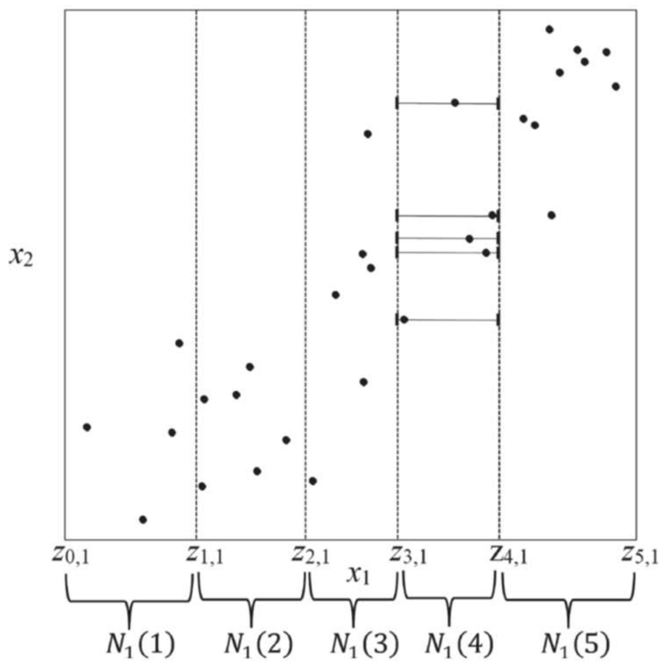
The centring ensures that $f_{j,ALE}(x_j)$ as mean 0 with respect to the marginal distribution of X_j.

Approximation

- The uncentered ALE is approximated by

$$\hat{g}_{j,ALE}(x) = \sum_{k=1}^{k_j(x)} \frac{1}{n_j(k)} \sum_{\{i: x_{i,j} \in N_j(k)\}} [f(z_{k,j}, \mathbf{x}_{i,\setminus j}) - f(z_{k-1,j}, \mathbf{x}_{i,\setminus j})]$$

- The sample range of variable x_j is split into *K* intervals with split points $z_{0,j}, \dots, z_{K,j}$.
- $n_{j(k)}$ is the number of training observations that falls into the kth interval $N_{j(k)}$.
- $k_{j(x)}$ is the index of the interval into which *x* falls.

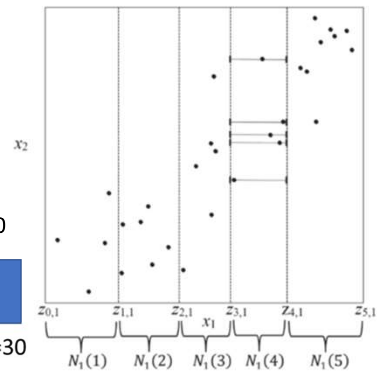


Example:

- 2 variables
- Compute ALE for X_1
- Range of X_1 is [0,50]
- 30 training observations
- K=5 intervals
- $n_1(1) = 4, n_1(2) = 6,\ n_1(3) = 6,\ n_1(4) = 5, n_1(5) = 9$

Source: Apley and Zhu (2006)

Example with n=30 and K=5



1…………………10 11…………………20 21…………………30 31…………………40 41…………… ..50

$n_1(1)$=4	$n_1(2)$=6	$n_1(3)$=6	$n_1(3)$=6	$n_1(3)$=6

$z_{0,1}$=1 $z_{1,1}$=7 $z_{2,1}$=13 $z_{5,1}$=30

Want to compute ALE for x1=11, which is in the second interval. Hence, $k_{1(x)}$ (interval into which x falls) is 2.
The outer sum for $g_{j(x)}$ consists of two terms, corresponding to k=1 and k=2.
We have that $n_1(1)$ is 4 and $n_1(2)$ is 6.
The inner sum of the first term runs over the 4 observations *i* for which $x_{i,1}$ is between $z_{0,1}$ and $z_{1,1}$, while the inner sum of the second term runs over the 6 observations *i* for which $x_{i,1}$ is between $z_{1,1}$ and $z_{2,1}$.

$$\hat{g}_{j,ALE}(x) = \sum_{k=1}^{k_j(x)} \frac{1}{n_j(k)} \sum_{\{i: x_{i,j} \in N_j(k)\}} [f(z_{k,j}, \mathbf{x}_{i,\setminus j}) - f(z_{k-1,j}, \mathbf{x}_{i,\setminus j})]$$

Centering

- The final ALE estimator is obtained by subtracting an estimate of $\mathbb{E}[g_{j,ALE}(X_j)]$ from the uncentered version, i.e.

$$\hat{f}_{j,ALE}(x) = \hat{g}_{j,ALE}(x) - \frac{1}{n}\sum_{i=1}^{n}\hat{g}_{j,ALE}(x_{i,j}) = \hat{g}_{j,ALE}(x) - \frac{1}{n}\sum_{k=1}^{K} n_j(k)\hat{g}_{j,ALE}(z_{k,j}).$$

- This means that $\hat{f}_{j,ALE}(x)$ can be interpreted as the main effect of feature *j* at value *x* compared to the average prediction of the data.
- If $\hat{f}_{j,ALE}(x)$ = -2 when *x*=3 it means that when feature *j* has value 3, then the prediction is equal to the average prediction minus 2.

Pros and Cons

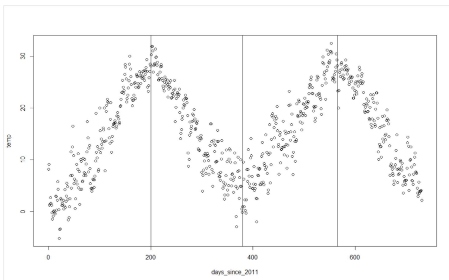
- Pros:
  - Handles correlated features
  - Are faster to compute than PDs
  - Generally easy to interpret
- Cons:
  - Not straightforward to handle categorical features (need an order).
  - Not straightforward how to determine the number of intervals.
  - The method is less intuitive compared to PD plots.
  - Interpretation remains difficult when features are strongly correlated.

Software

- PYTHON:
  - ALEPython package: https://github.com/blent-ai/ALEPython
- R:
  - ALEPlot R package: https://cran.r-project.org/web/packages/ALEPlot/index.html
  - iml R package: https://cran.r-project.org/web/packages/iml/index.html

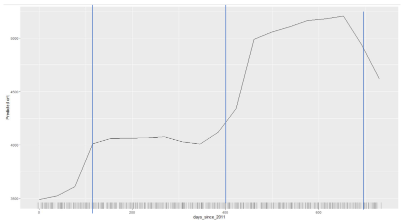
Examples

- Bike data set
- Fit a Random forest model with 50 trees
- temp and days_since_2011 are dependent:
  - High values for temp when days_since_2011 are around 200 and 565 (summer)
  - Low values for temp when days_since_2011 are around 20 and 380 (winter)
- PD plot uses all combinations of the two variables.

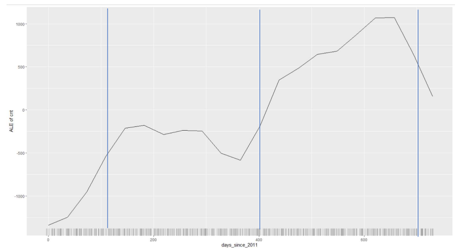


PD and ALE plots: days_since_2011

PD-plot



ALE-plot

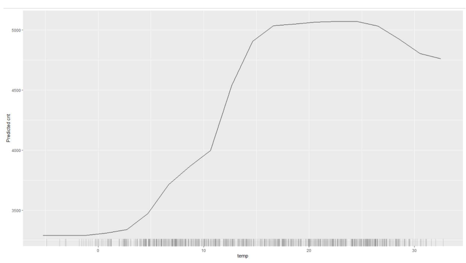


```
library("iml")
library("randomForest")
library("patchwork")

load("H://NTNU//ExplainableAI/bike.Rdata")
rf <- randomForest(cnt ~ ., data = bike, ntree = 50)
print(rf)

mod <- Predictor$new(rf, data = bike)
```

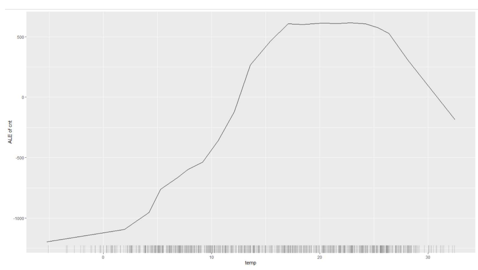
```
eff <- FeatureEffect$new(mod, feature = "days_since_2011", method="ale")
plot(eff)
eff <- FeatureEffect$new(mod, feature = "days_since_2011", method="pdp")
plot(eff)
```

PD and ALE plots : temp

PD-plot

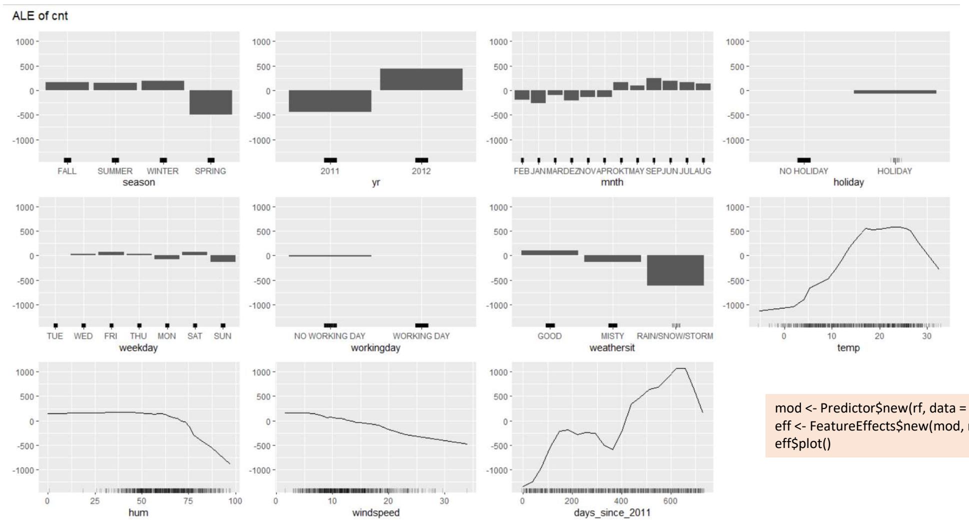


ALE-plot



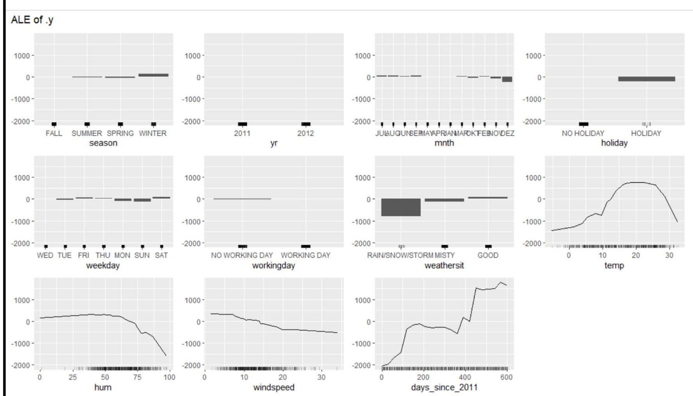
```
#PD plot
eff <- FeatureEffect$new(mod, feature = "temp", method="pdp")
plot(eff)

#ALE plot
eff <- FeatureEffect$new(mod, feature = "temp", method="ale")
plot(eff)
```

Random forest ALE




```
mod <- Predictor$new(rf, data = bike)
eff <- FeatureEffects$new(mod, method="ale")
eff$plot()
```

XGBoost ALE




```
# 1. create a data frame with just the features
features <- bikeTrain[,-11]

# 2. Create a vector with the actual responses
response <- bikeTrain[,"cnt"]

# 3. Create custom predict function that returns the predicted values as a  vector
pred <- function(model, newdata)
{
  #xgb.test  <- xgb.DMatrix(data = as.matrix(sapply(newdata[,-11], as.numeric)),label = newdata[,11])
  xgb.test  <- xgb.DMatrix(data = as.matrix(sapply(newdata, as.numeric)))
  results <- predict(model,newdata=xgb.test)
  #return(results[[3L]])
  return(results)
}

#4. Define predictor
predictor.xgb <- Predictor$new(
  model = model.xgb,
  data = features,
  y = response,
  predict.fun = pred,
  class = "regression"
)

#5. Compute feature effects
eff <- FeatureEffect$new(predictor.xgb, feature = "temp", method="ale")
plot(eff)
eff <- FeatureEffects$new(predictor.xgb, method="ale")
eff$plot()
```