MA8701 Advanced methods in statistical inference and learning

L18 with Kjersti Aas

Mette Langaas IMF/NTNU

20 March, 2023

Contents

Part 4: Explainable AI

The main role of this note is to give the R code for the examples found on the Part 4 slide sets, and detailed references for further reading.

Why a part on XAI?

In **Part 2** we worked with *interpretable* methods:

- linear regression (LS/MLE, ridge and lasso)
- logistic regression (MLE, ridge and lasso)

By studying the estimated regression coefficients we could (to some extent) explain what our fitted model could tell us about the data we had analysed.

In **Part 3** we started by studying a classification and regression tree, which is also an interpretable method, but also different versions of ensemble methods (bagging, random forest, xgboost, superlearner) - which are not interpretable.

We may refer to the methods of Part 3 as *black box* methods, since in a prediction setting we would input an observation to the fitted method and the method would output a prediction - but we would not have a specific formula that we use to explain why the method gave this prediction.

In many situations we would like to know more about the model that the method have fitted. We would like some kind of interpretation of what the underlying methods does, for example:

- what is the mathematical relationship between x and y in the fitted method?
- how much of the variability in the data is explained by feature x in the fitted method?

- is there an interaction effect between x_1 and x_2 in the fitted method?

Remark: we want to interpret the fitted method, based on the available data (but not really interpret directly the data).

We would also like to *explain* the prediction for a given input.

See Chapter 3 of Molnar (2023) on a discussion of *interpretability*.

Reading list

- Molnar (2023): Chapters 3, 6, 8 (not 8,3, 8.4,8.6,8.7, 9 (not 9.4, 9.6.3) from https://christophm.github.io/interpretable-ml-book/
- Three slide sets from Kjersti Aas (on Blackboard)
  - Introduction
  - LIME and counterfactual explantations
  - Shapley values

Supplementary reading is specified for (below).

Outline

We start by motivating the need for XAI, and then look at

- Global explanation methods
  - Model specific methods
  - Model agnostic methods (PDP plots, ICE plots, ALE plots)
- Local explanation methods
  - Method specific
  - Model agnostic (LIME, Shapley values, Counterfactual explanations)

L18: Introduction slide set

Analysis of the bike data

Linear model

```r
# download manually
#"https://github.com/christophM/interpretable-ml-book/blob/master/data/bike.Rdata"

load("bike.Rdata")
colnames(bike)
```

```
##  [1] "season"         "yr"             "mnth"           "holiday"
##  [5] "weekday"        "workingday"     "weathersit"     "temp"
##  [9] "hum"            "windspeed"      "cnt"            "days_since_2011"
```

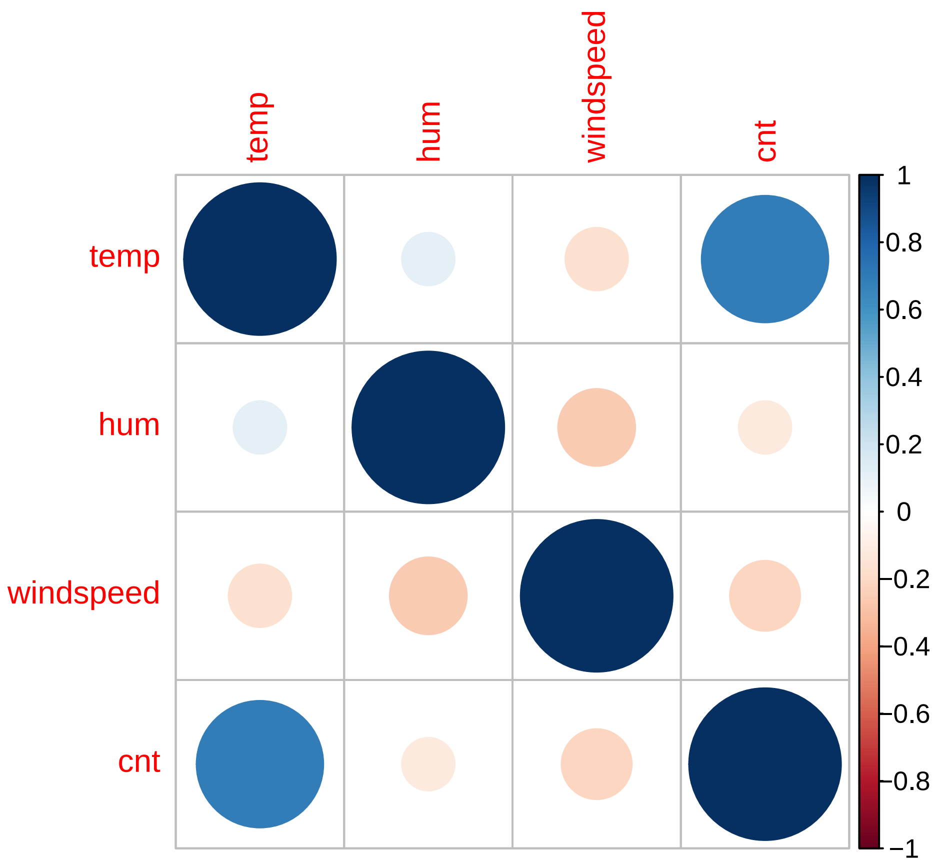
```r
n=dim(bike)[1]
bikeTrain=bike[1:600,]
bikeTest<-bike[601:n,]

linearMod <- lm(cnt~.,data=bikeTrain) #bikeTrain

tmp <- summary(linearMod)
tmp$r.square
```

```
## [1] 0.8609615
```

```r
tmp$coefficients[rev(order(abs(tmp$coefficients[,3]))),]
```

```
##                                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
## seasonFALL                  1022.757860  219.109652  4.6677901 3.797994e-06
## hum                          -12.557802    2.840354 -4.4212094 1.175172e-05
## seasonSPRING                 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
## seasonSUMMER                 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
## mnthOCT                     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
## mnthDEC                     1104.438624 1128.543616  0.9786406 3.281720e-01
## days_since_2011               -1.257864    3.357210 -0.3746754 7.080410e-01
```

```r
corrplot(cor(bikeTrain[,8:11]))
```



LMG

Problems with variable 6, removed for the LMG-method.

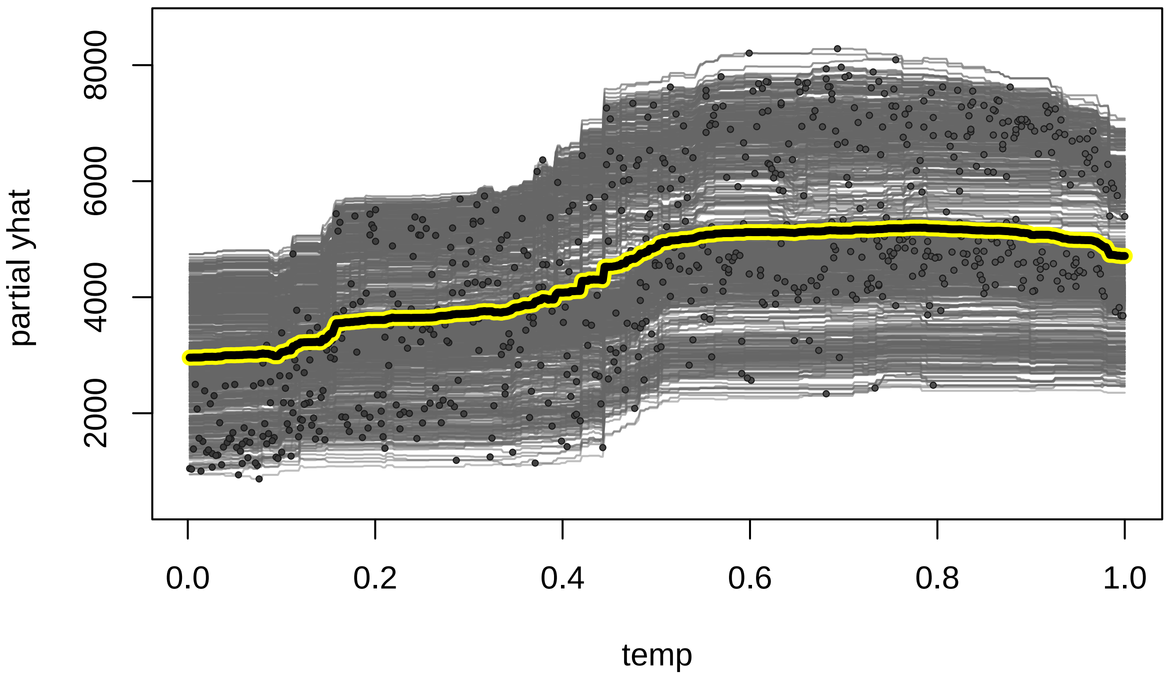
```
library("relaimpo")
calc.relimp(cnt~., data=bikeTrain[,-6], type="lmg",rela=TRUE)
rev(sort(crf$lmg))
```

ALE and PDP for RF

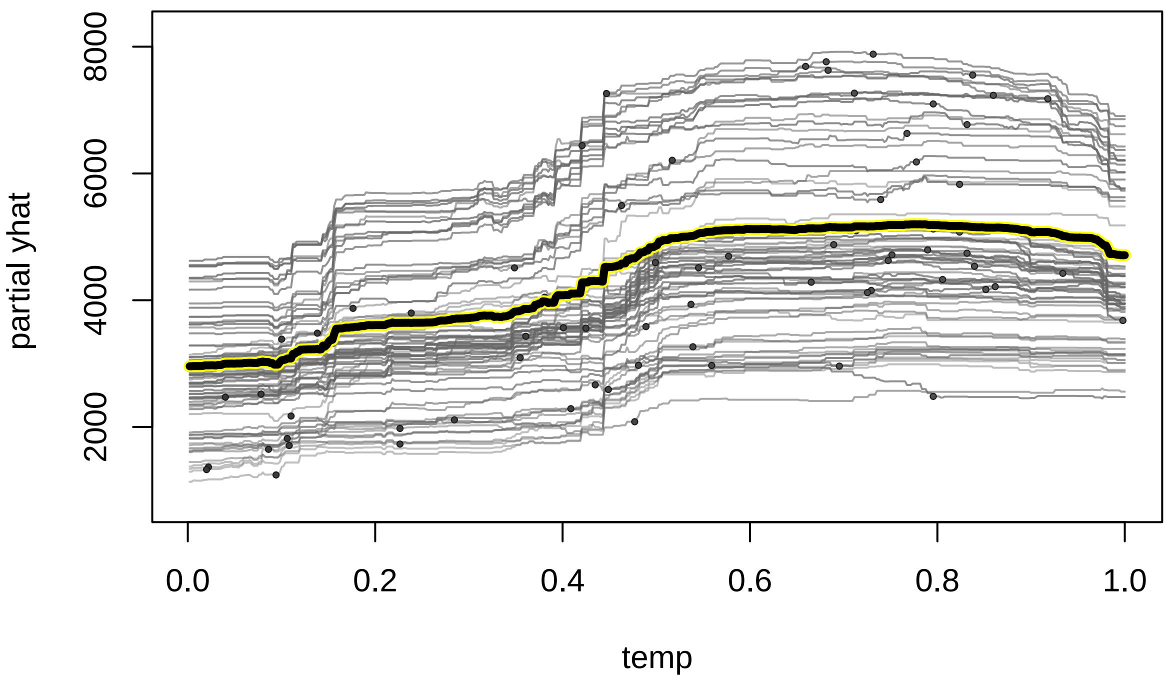
```
# ICE
X=model.matrix(~.-cnt,data=bike)
rf=randomForest(y=bike$cnt, x=X,ntree=50,  importance=TRUE)
this=ice(rf,X=X,predictor=27,plot=TRUE)
```

```
## ..........................................................................................
## y not passed, so range_y is range of ice curves and sd_y is sd of predictions on real observations
```

```
plot(this,centered=FALSE,xlab="temp",frac_to_plot=1,plot_orig_pts_preds=TRUE,pts_preds_size=0.5)
```




```
plot(this,centered=FALSE,xlab="temp",frac_to_plot=0.1,plot_orig_pts_preds=TRUE,pts_preds_size=0.5)
```

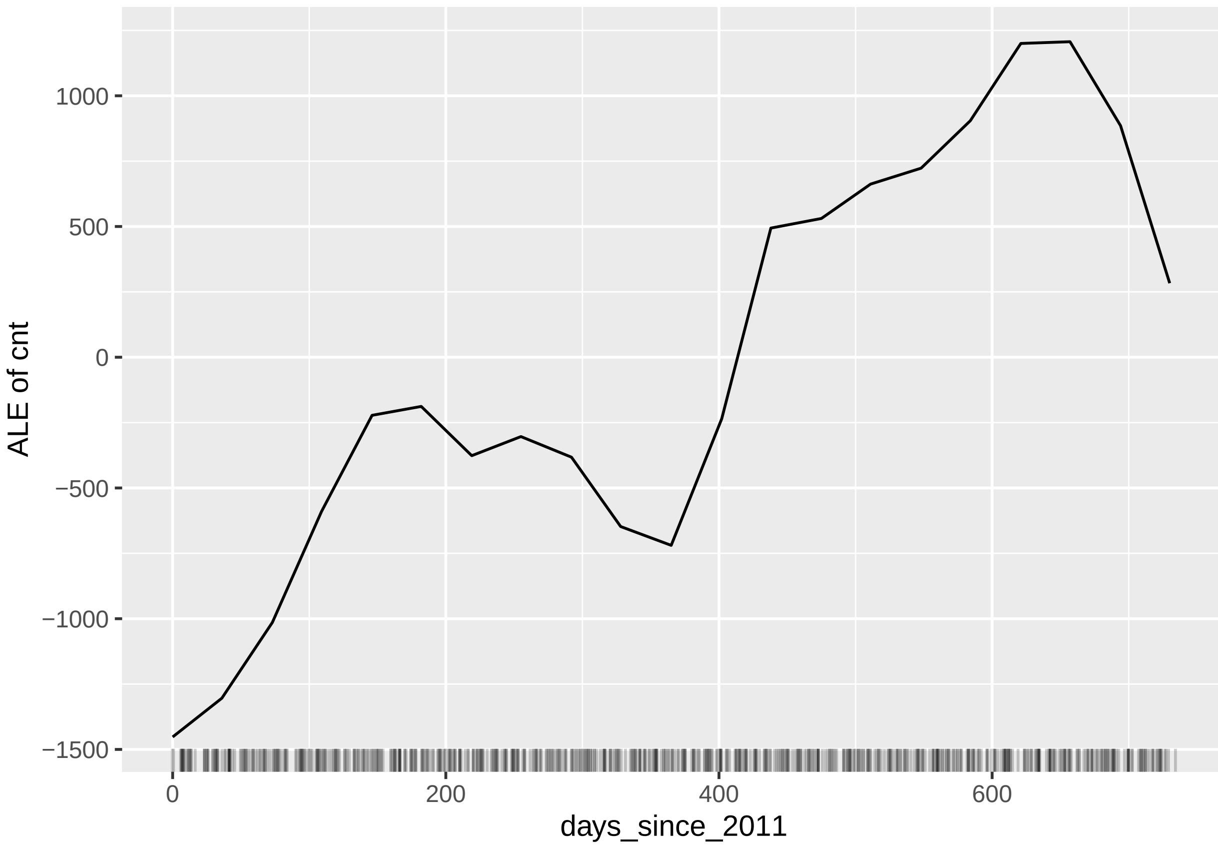



```
rf=randomForest(cnt ~ ., data = bike, ntree = 50)
print(rf)
```

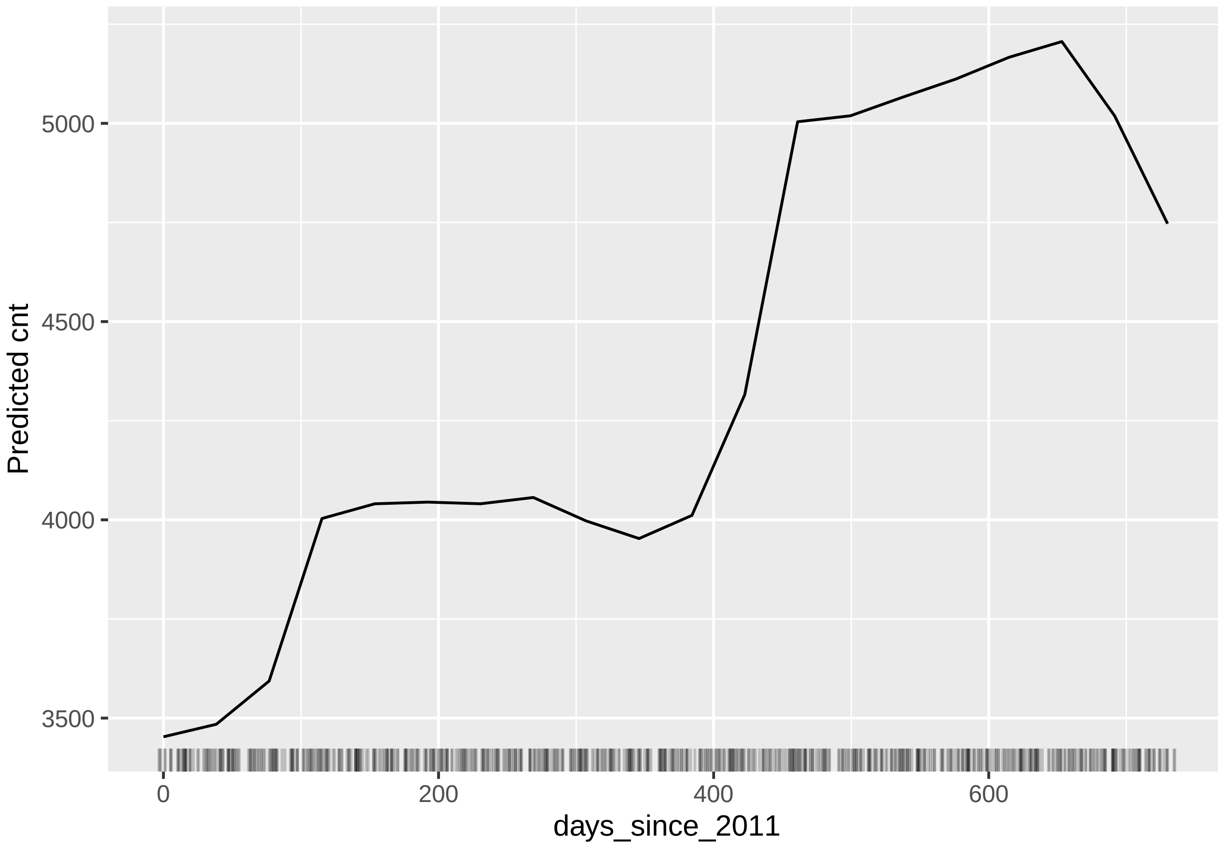
```
##
## Call:
##  randomForest(formula = cnt ~ ., data = bike, ntree = 50)
##                Type of random forest: regression
##                      Number of trees: 50
## No. of variables tried at each split: 3
```

```
##
##           Mean of squared residuals: 450920.9
##                     % Var explained: 87.97
```

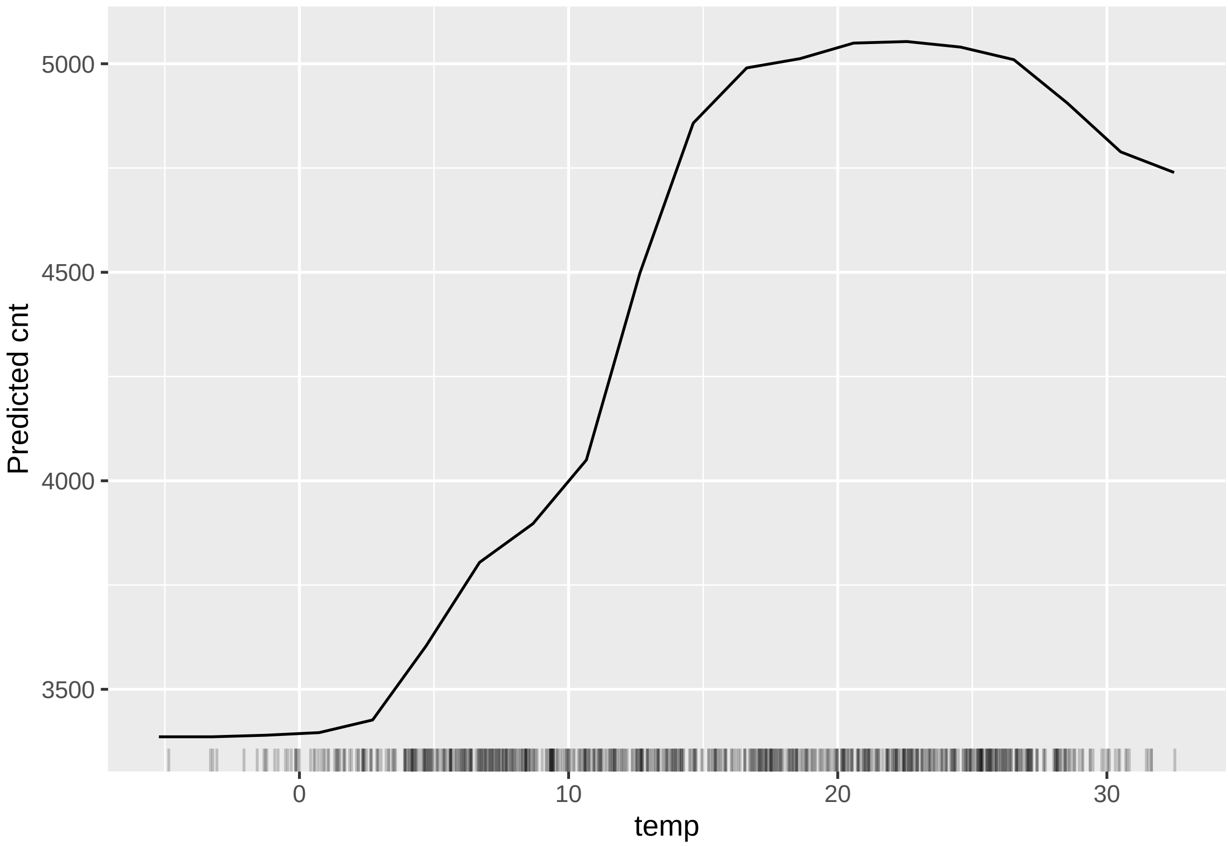
```r
mod=Predictor$new(rf, data = bike)


eff1=FeatureEffect$new(mod, feature = "days_since_2011", method="ale")
plot(eff1)
```



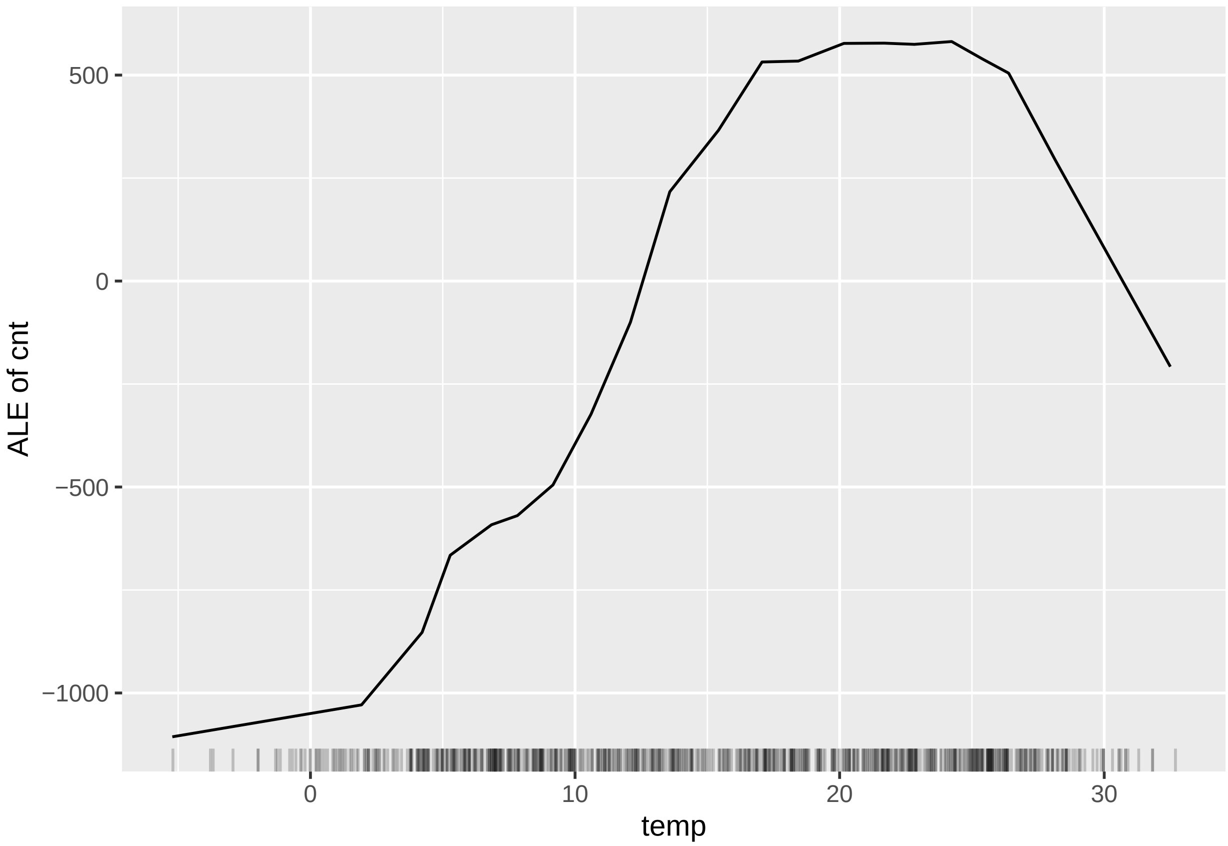

```r
eff2=FeatureEffect$new(mod, feature = "days_since_2011", method="pdp")
plot(eff2)
```



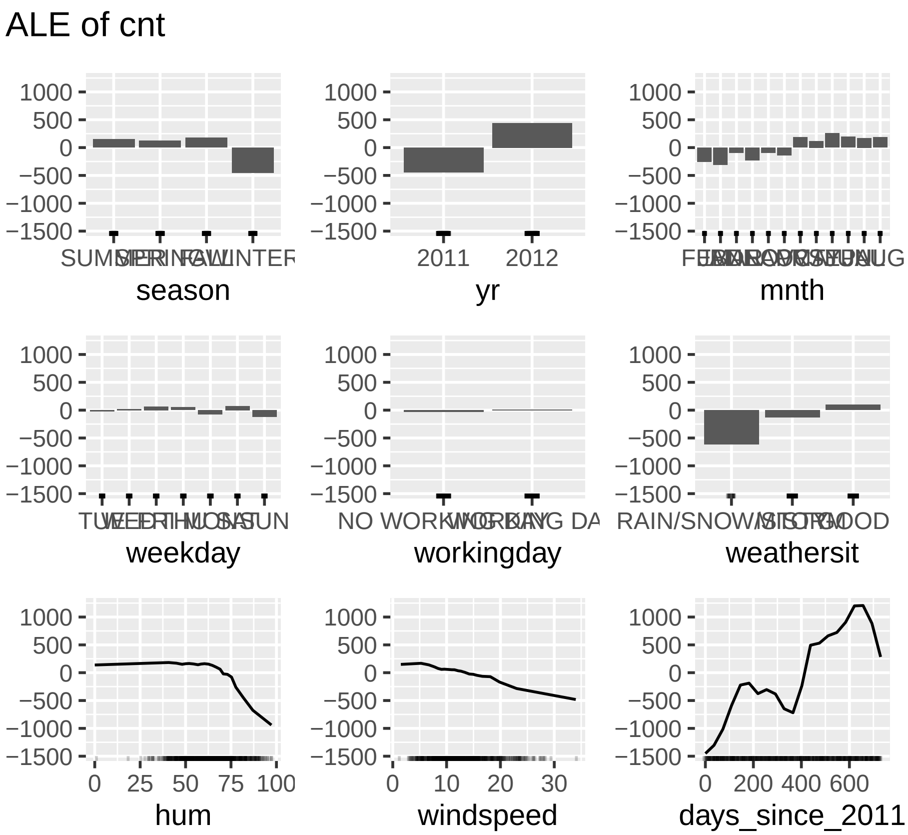

```r
#PD plot
eff1=FeatureEffect$new(mod, feature = "temp", method="pdp")
plot(eff1)
```




```r
#ALE plot
eff2=FeatureEffect$new(mod, feature = "temp", method="ale")
plot(eff2)
```




```r
eff<-FeatureEffects$new(mod, method="ale")
eff$plot()
```




```
#eff<-FeatureEffects$new(mod, method="ice",feature="temp")
#eff$plot()
```

ALE and PDP for xgboost

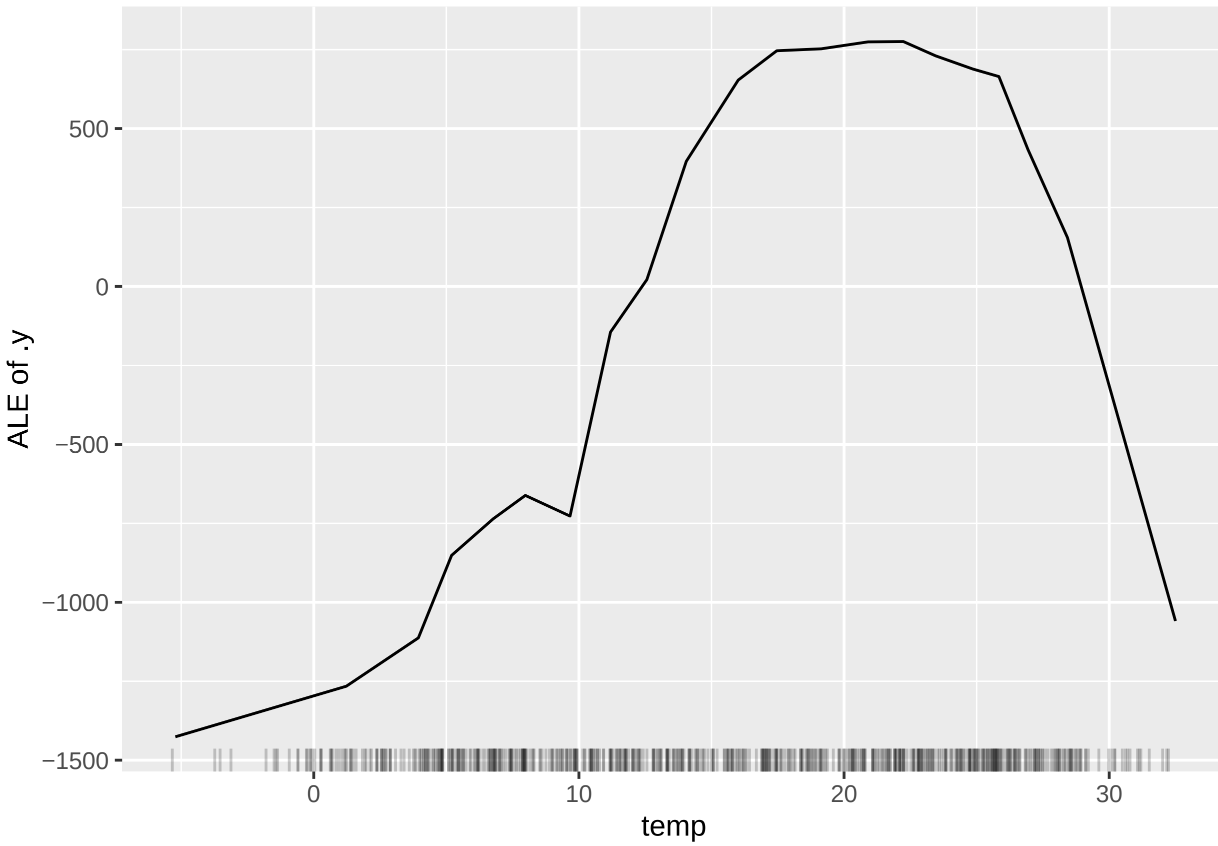
```
library(xgboost)
n<-dim(bike)[1]
bikeTrain<-bike[1:600,]
bikeTest<-bike[601:n,]
xgb.train=xgb.DMatrix(data = as.matrix(sapply(bikeTrain[,-11], as.numeric)),label = bikeTrain[,"cnt"])
xgb.test<-xgb.DMatrix(data = as.matrix(sapply(bikeTest[,-11], as.numeric)),label = bikeTest[,"cnt"])

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)
```

```
## [20:05:24] WARNING: src/learner.cc:767:
## Parameters: { "ntread" } are not used.
##
## [1]  train-rmse:4157.409393  test-rmse:5570.701759
## [11] train-rmse:1558.593197  test-rmse:2442.684107
## [21] train-rmse:648.676119   test-rmse:1493.601085
## [31] train-rmse:330.253573   test-rmse:1226.610739
## [41] train-rmse:215.791169   test-rmse:1151.691631
## [50] train-rmse:172.685072   test-rmse:1135.281199
```

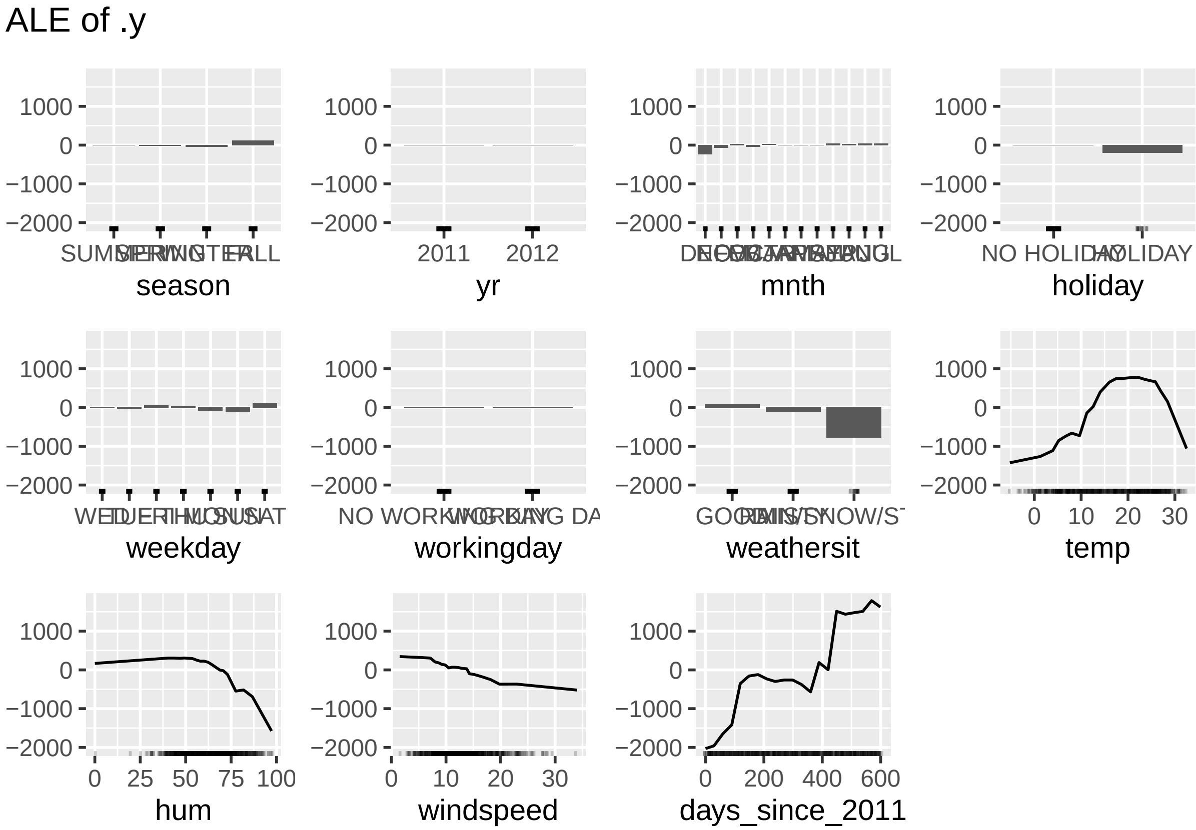
```r
xgb.importance(model=model)
```

```
##             Feature         Gain        Cover   Frequency
##  1: days_since_2011 0.7976585761 0.3430909388 0.162966462
##  2:            temp 0.1132750493 0.2445233763 0.239017478
##  3:             hum 0.0295591154 0.1416631672 0.180444025
##  4:      weathersit 0.0213786713 0.0364816629 0.049598488
##  5:       windspeed 0.0192767296 0.0998856850 0.145488899
##  6:         weekday 0.0055235514 0.0546554218 0.091639112
##  7:            mnth 0.0054332987 0.0396486562 0.056683987
##  8:          season 0.0031174009 0.0115831467 0.024090694
##  9:      workingday 0.0023060853 0.0175030328 0.033065659
## 10:         holiday 0.0021865774 0.0105916387 0.013698630
## 11:              yr 0.0002849445 0.0003732736 0.003306566
```

```r
# 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$new(
model = model,
data = features,
y = response,
predict.fun = pred,
class = "regression"
)
#5. Compute feature effects
eff<-FeatureEffect$new(predictor.xgb, feature = "temp", method="ale")
plot(eff)
```




```
eff<-FeatureEffects$new(predictor.xgb, method="ale")
eff$plot()
```



References for further reading

- LMG: the method is nicely explained in Grömping (2007)
- ICEplot: Goldstein et al. (2015)
- ALEplot: Apley and Zhu (2020)
- PDPplot: Friedman, Hastie, and Tibshirani (2001) chapter 10.13.2

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/https://doi.org/10.1111/rssb.12377.

Friedman, Jerome, Trevor Hastie, and Robert Tibshirani. 2001. *The Elements of Statistical Learning*. Vol. 1. Springer series in statistics New York.

Goldstein, Alex, Adam Kapelner, Justin Bleich, and Emil Pitkin. 2015. "Peeking Inside the Black Box: Visualizing Statistical Learning with Plots of Individual Conditional Expectation." *Journal of Computational and Graphical Statistics* 24 (1): 44–65. https://doi.org/10.1080/10618600.2014.907095.

Grömping, U. 2007. "Estimators of Relative Importance in Linear Regression Based on Variance Decomposition." *The American Statistician* 61: 139–47.

Molnar, Christoph. 2023. *Interpretable Machine Learning: A Guide for Making Black Box Models Explainable*.