Package 'CAST'

July 17, 2020

Type Package

Title 'caret' Applications for Spatial-Temporal Models

Version 0.4.2

Author Hanna Meyer [cre, aut],
Chris Reudenbach [ctb],
Marvin Ludwig [ctb],
Thomas Nauss [ctb],
Edzer Pebesma [ctb]

Maintainer Hanna Meyer <hanna.meyer@uni-muenster.de>

Description Supporting functionality to run 'caret' with spatial or spatial-temporal data. 'caret' is a frequently used package for model training and prediction using machine learning. This package includes functions to improve spatial-temporal modelling tasks using 'caret'. It prepares data for Leave-Location-Out and Leave-Time-Out cross-validation which are target-oriented validation strategies for spatial-temporal models. To decrease overfitting and improve model performances, the package implements a forward feature selection that selects suitable predictor variables in view to their contribution to the target-oriented performance.

License GPL (>= 3) | file LICENSE

URL https://github.com/HannaMeyer/CAST

Encoding UTF-8

LazyData true

Depends R (>= 3.1.0)

Imports caret, stats, utils, ggplot2, graphics, reshape, FNN, plyr

Suggests doParallel, GSIF, randomForest, lubridate, raster, sp, knitr, mapview, rmarkdown, sf, scales, parallel, latticeExtra, virtualspecies, gridExtra, viridis, rgeos

RoxygenNote 7.1.0

VignetteBuilder knitr

NeedsCompilation no

Repository CRAN

Date/Publication 2020-07-17 12:32:13 UTC

R topics documented:

---

aoa *Area of Applicability*

Description

This function estimates the Dissimilarity Index (DI) and the derived Area of Applicability (AOA) of spatial prediction models by considering the distance of new data (i.e. a Raster Stack of spatial predictors used in the models) in the predictor variable space to the data used for model training. Predictors can be weighted in the ideal case based on the internal variable importance of the machine learning algorithm used for model training.

Usage

```
aoa(
  newdata,
  model = NA,
  train = NULL,
  weight = NA,
  variables = "all",
  thres = 0.95,
  folds = NULL
)
```

Arguments

newdata	A RasterStack, RasterBrick or data.frame containing the data the model was meant to make predictions for.
model	A train object created with caret used to extract weights from (based on variable importance) as well as cross-validation folds
train	a data.frame containing the data used for model training. Only required when no model is given
weight	A data.frame containing weights for each variable. Only required if no model is given.
variables	character vector of predictor variables. if "all" then all variables of the model are used or if no model is given then of the train dataset.
thres	numeric vector of probability of DI in training data, with values in [0,1].

folds — Numeric or character. Folds for cross validation. E.g. Spatial cluster affiliation for each data point. Should be used if replicates are present. Only required if no model is given.

Details

The Dissimilarity Index (DI) and the corresponding Area of Applicability (AOA) are calculated. If variables are factors, dummy variables are created prior to weighting and distance calculation.

Interpretation of results: If a location is very similar to the properties of the training data it will have a low distance in the predictor variable space (DI towards 0) while locations that are very different in their properties will have a high DI. To get the AOA, a threshold to the DI is applied based on the DI in the training data. To calculate the DI in the training data, the minimum distance to an other training point (if applicable: not located in the same CV fold) is considered. See Meyer and Pebesma (2020) for the full documentation of the methodology.

Value

A RasterStack or data.frame with the DI and AOA. AOA has values 0 (outside AOA) and 1 (inside AOA).

Note

Note extensively tested with categorical predictors yet!

Author(s)

Hanna Meyer

References

Meyer, H., Pebesma, E. (2020): Predicting into unknown space? Estimating the area of applicability of spatial prediction models. https://arxiv.org/abs/2005.07939

Examples

```
## Not run:
library(sf)
library(raster)
library(caret)
library(viridis)
library(latticeExtra)

# prepare sample data:
dat <- get(load(system.file("extdata","Cookfarm.RData",package="CAST")))
dat <- aggregate(dat[,c("VW","Easting","Northing")],by=list(as.character(dat$SOURCEID)),mean)
pts <- st_as_sf(dat,coords=c("Easting","Northing"))
pts$ID <- 1:nrow(pts)
set.seed(100)
pts <- pts[1:30,]
studyArea <- stack(system.file("extdata","predictors_2012-03-25.grd",package="CAST"))[[1:8]]
trainDat <- extract(studyArea,pts,df=TRUE)
```

```
trainDat <- merge(trainDat,pts,by.x="ID",by.y="ID")

# visualize data spatially:
spplot(scale(studyArea))
plot(studyArea$DEM)
plot(pts[,1],add=TRUE,col="black")

# train a model:
set.seed(100)
variables <- c("DEM","NDRE.Sd","TWI")
model <- train(trainDat[,which(names(trainDat)%in%variables)],
trainDat$VW,method="rf",importance=TRUE,tuneLength=1,trControl=trainControl(method="cv",number=5))
print(model) #note that this is a quite poor prediction model
prediction <- predict(studyArea,model)
plot(varImp(model,scale=FALSE))

#...then calculate the AOA of the trained model for the study area:
AOA <- aoa(studyArea,model)
spplot(AOA$DI, col.regions=viridis(100),main="Applicability Index")
#plot predictions for the AOA only:
spplot(prediction, col.regions=viridis(100),main="prediction for the AOA")+
spplot(AOA$AOA,col.regions=c("grey","transparent"))

#The AOA can also be calculated without a trained model.
#All variables are weighted equally in this case:
AOA <- aoa(studyArea,train=trainDat,variables=variables)
spplot(AOA$DI, col.regions=viridis(100),main="Applicability Index")
spplot(AOA$AOA,main="Area of Applicability")

## End(Not run)
```

---

bss *Best subset feature selection*

Description

Evaluate all combinations of predictors during model training

Usage

```
bss(
  predictors,
  response,
  method = "rf",
  metric = ifelse(is.factor(response), "Accuracy", "RMSE"),
  maximize = ifelse(metric == "RMSE", FALSE, TRUE),
  trControl = caret::trainControl(),
  tuneLength = 3,
  tuneGrid = NULL,
```

```
  seed = 100,
  verbose = TRUE,
  ...
)
```

Arguments

predictors	see train
response	see train
method	see train
metric	see train
maximize	see train
trControl	see train
tuneLength	see train
tuneGrid	see train
seed	A random number
verbose	Logical. Should information about the progress be printed?
...	arguments passed to the classification or regression routine (such as randomForest).

Details

bss is an alternative to ffs and ideal if the training set is small. Models are iteratively fitted using all different combinations of predictor variables. Hence, 2^X models are calculated. Dont try running bss on very large datasets because the computation time is much higher compared to ffs.

The internal cross validation can be run in parallel. See information on parallel processing of carets train functions for details.

Value

A list of class train. Beside of the usual train content the object contains the vector "selectedvars" and "selectedvars_perf" that give the best variables selected as well as their corresponding performance. It also contains "perf_all" that gives the performance of all model runs.

Note

This validation is particulary suitable for spatial leave-location-out cross validations where variable selection MUST be based on the performance of the model on the hold out station. Note that bss is very slow since all combinations of variables are tested. A more time efficient alternative is the forward feature selection (ffs) (ffs).

Author(s)

Hanna Meyer

See Also

train,ffs, trainControl,CreateSpacetimeFolds

Examples

```
## Not run:
data(iris)
bssmodel <- bss(iris[,1:4],iris$Species)
bssmodel$perf_all

## End(Not run)
```

CAST *'caret' Applications for Spatial-Temporal Models*

Description

Supporting functionality to run 'caret' with spatial or spatial-temporal data. 'caret' is a frequently used package for model training and prediction using machine learning. CAST includes functions to improve spatial-temporal modelling tasks using 'caret'. It supports Leave-Location-Out and Leave-Time-Out cross-validation of spatial and spatial-temporal models and allows for spatial variable selection to selects suitable predictor variables in view to their contribution to the spatial model performance. CAST further includes functionality to estimate the (spatial) area of applicability of prediction models by analysing the similarity between new data and training data.

Details

'caret' Applications for Spatio-Temporal models

Author(s)

Hanna Meyer

References

- Meyer, H., Pebesma, E. (2020): Predicting into unknown space? Estimating the area of applicability of spatial prediction models. https://arxiv.org/abs/2005.07939.
- Meyer, H., Reudenbach, C., Wöllauer, S., Nauss, T. (2019): Importance of spatial predictor variable selection in machine learning applications - Moving from data reproduction to spatial prediction. Ecological Modelling. 411, 108815.
- Meyer, H., Reudenbach, C., Hengl, T., Katurji, M., Nauß, T. (2018): Improving performance of spatio-temporal machine learning models using forward feature selection and target-oriented validation. Environmental Modelling & Software 101: 1-9.

`CreateSpacetimeFolds` *Create Space-time Folds*

Description

Create spatial, temporal or spatio-temporal Folds for cross validation

Usage

```
CreateSpacetimeFolds(
  x,
  spacevar = NA,
  timevar = NA,
  k = 10,
  class = NA,
  seed = sample(1:1000, 1)
)
```

Arguments

`x`	data.frame containing spatio-temporal data
`spacevar`	Character indicating which column of x identifies the spatial units (e.g. ID of weather stations)
`timevar`	Character indicating which column of x identifies the temporal units (e.g. the day of the year)
`k`	numeric. Number of folds. If spacevar or timevar is NA and a leave one location out or leave one time step out cv should be performed, set k to the number of unique spatial or temporal units.
`class`	Character indicating which column of x identifies a class unit (e.g. land cover)
`seed`	numeric. See ?seed

Details

Using "class" is helpful in the case that data are clustered in space and are categorical. E.g This is the case for land cover classifications when training data come as training polygons. In this case the data should be split in a way that entire polygons are held back (spacevar="polygonID") but at the same time the distribution of classes should be similar in each fold (class="LUC").

Value

A list that contains a list for model training and a list for model validation that can directly be used as "index" and "indexOut" in caret's trainControl function

Note

Standard k-fold cross-validation can lead to considerable misinterpretation in spatial-temporal modelling tasks. This function can be used to prepare a Leave-Location-Out, Leave-Time-Out or Leave-Location-and-Time-Out cross-validation as target-oriented validation strategies for spatial-temporal prediction tasks. See Meyer et al. (2018) for further information.

Author(s)

Hanna Meyer

References

Meyer, H., Reudenbach, C., Hengl, T., Katurji, M., Nauß, T. (2018): Improving performance of spatio-temporal machine learning models using forward feature selection and target-oriented validation. Environmental Modelling & Software 101: 1-9.

See Also

trainControl,ffs

Examples

```
library(GSIF)
data(cookfarm)
### Prepare for 10-fold Leave-Location-and-Time-Out cross validation
indices <- CreateSpacetimeFolds(cookfarm$readings,"SOURCEID","Date")
str(indices)
### Prepare for 10-fold Leave-Location-Out cross validation
indices <- CreateSpacetimeFolds(cookfarm$readings,spacevar="SOURCEID")
str(indices)
### Prepare for leave-One-Location-Out cross validation
indices <- CreateSpacetimeFolds(cookfarm$readings,spacevar="SOURCEID",
k=length(unique(cookfarm$readings$SOURCEID)))
str(indices)
```

---

ffs *Forward feature selection*

Description

A simple forward feature selection algorithm

Usage

```
ffs(
  predictors,
  response,
  method = "rf",
```

```
  metric = ifelse(is.factor(response), "Accuracy", "RMSE"),
  maximize = ifelse(metric == "RMSE", FALSE, TRUE),
  withinSE = FALSE,
  minVar = 2,
  trControl = caret::trainControl(),
  tuneLength = 3,
  tuneGrid = NULL,
  seed = sample(1:1000, 1),
  verbose = TRUE,
  ...
)
```

Arguments

predictors	see train
response	see train
method	see train
metric	see train
maximize	see train
withinSE	Logical Models are only selected if they are better than the currently best models Standard error
minVar	Numeric. Number of variables to combine for the first selection. See Details.
trControl	see train
tuneLength	see train
tuneGrid	see train
seed	A random number used for model training
verbose	Logical. Should information about the progress be printed?
...	arguments passed to the classification or regression routine (such as randomForest).

Details

Models with two predictors are first trained using all possible pairs of predictor variables. The best model of these initial models is kept. On the basis of this best model the predictor variables are iteratively increased and each of the remaining variables is tested for its improvement of the currently best model. The process stops if none of the remaining variables increases the model performance when added to the current best model.

The internal cross validation can be run in parallel. See information on parallel processing of carets train functions for details.

Using withinSE will favour models with less variables and probably shorten the calculation time

Per Default, the ffs starts with all possible 2-pair combinations. minVar allows to start the selection with more than 2 variables, e.g. minVar=3 starts the ffs testing all combinations of 3 (instead of 2) variables first and then increasing the number. This is important for e.g. neural networks that often cannot make sense of only two variables. It is also relevant if it is assumed that the optimal variables can only be found if more than 2 are considered at the same time.

Value

A list of class train. Beside of the usual train content the object contains the vector "selectedvars" and "selectedvars_perf" that give the order of the best variables selected as well as their corresponding performance (starting from the first two variables). It also contains "perf_all" that gives the performance of all model runs.

Note

This validation is particulary suitable for spatial leave-location-out cross validations where variable selection MUST be based on the performance of the model on the hold out station. See Meyer et al. (2018) and Meyer et al. (2019) for further details.

Author(s)

Hanna Meyer

References

- Gasch, C.K., Hengl, T., Gräler, B., Meyer, H., Magney, T., Brown, D.J. (2015): Spatio-temporal interpolation of soil water, temperature, and electrical conductivity in 3D+T: the Cook Agronomy Farm data set. Spatial Statistics 14: 70-90.
- Meyer, H., Reudenbach, C., Hengl, T., Katurji, M., Nauß, T. (2018): Improving performance of spatio-temporal machine learning models using forward feature selection and target-oriented validation. Environmental Modelling & Software 101: 1-9.
- Meyer, H., Reudenbach, C., Wöllauer, S., Nauss, T. (2019): Importance of spatial predictor variable selection in machine learning applications - Moving from data reproduction to spatial prediction. Ecological Modelling. 411, 108815.

See Also

train,bss, trainControl,CreateSpacetimeFolds

Examples

```
## Not run:
data(iris)
ffsmodel <- ffs(iris[,1:4],iris$Species)
ffsmodel$selectedvars
ffsmodel$selectedvars_perf

## End(Not run)

# or perform model with target-oriented validation (LLO CV)
#the example is taken from the GSIF package and is described
#in Gasch et al. (2015). The ffs approach for this dataset is described in
#Meyer et al. (2018). Due to high computation time needed, only a small and thus not robust example
#is shown here.

## Not run:
#run the model on three cores:
```

```
library(doParallel)
cl <- makeCluster(3)
registerDoParallel(cl)

#load and prepare dataset:
dat <- get(load(system.file("extdata","Cookfarm.RData",package="CAST")))
trainDat <- dat[dat$altitude==-0.3&year(dat$Date)==2012&week(dat$Date)%in%c(13:14),]

#visualize dataset:
ggplot(data = trainDat, aes(x=Date, y=VW)) + geom_line(aes(colour=SOURCEID))

#create folds for Leave Location Out Cross Validation:
set.seed(10)
indices <- CreateSpacetimeFolds(trainDat,spacevar = "SOURCEID",k=3)
ctrl <- trainControl(method="cv",index = indices$index)

#define potential predictors:
predictors <- c("DEM","TWI","BLD","Precip_cum","cday","MaxT_wrcc",
"Precip_wrcc","NDRE.M","Bt","MinT_wrcc","Northing","Easting")

#run ffs model with Leave Location out CV
set.seed(10)
ffsmodel <- ffs(trainDat[,predictors],trainDat$VW,method="rf",
tuneLength=1,trControl=ctrl)
ffsmodel

#compare to model without ffs:
model <- train(trainDat[,predictors],trainDat$VW,method="rf",
tuneLength=1, trControl=ctrl)
model
stopCluster(cl)

## End(Not run)
```

`plot_ffs` *Plot results of a Forward feature selection or best subset selection*

Description

A plotting function for a forward feature selection result. Each point is the mean performance of a model run. Error bars represent the standard errors from cross validation. Marked points show the best model from each number of variables until a further variable could not improve the results. If type=="selected", the contribution of the selected variables to the model performance is shown.

Usage

```
plot_ffs(
  ffs_model,
  plotType = "all",
  palette = rainbow,
```

```
  reverse = FALSE,
  marker = "black",
  size = 1.5,
  lwd = 0.5,
  pch = 21,
  ...
)
```

Arguments

ffs_model	Result of a forward feature selection see ffs
plotType	character. Either "all" or "selected"
palette	A color palette
reverse	Character. Should the palette be reversed?
marker	Character. Color to mark the best models
size	Numeric. Size of the points
lwd	Numeric. Width of the error bars
pch	Numeric. Type of point marking the best models
...	Further arguments for base plot if type="selected"

Author(s)

Marvin Ludwig and Hanna Meyer

See Also

ffs, bss

Examples

```
## Not run:
data(iris)
ffsmodel <- ffs(iris[,1:4],iris$Species)
plot_ffs(ffsmodel)
#plot performance of selected variables only:
plot_ffs(ffsmodel,plotType="selected")

## End(Not run)
```

Index