# Package 'distillML'

July 7, 2022

**Type** Package

**Title** Model Distillation and Interpretability Methods for Machine Learning Models

**Version** 0.1.0.9

**Maintainer** Theo Saarinen <theo_s@berkeley.edu>

**BugReports** https://github.com/forestry-labs/distillML/issues

**URL** https://github.com/forestry-labs/distillML

**Description** Provides several methods for model distillation and interpretability for general black box machine learning models and treatment effect estimation methods. For details on the algorithms implemented, see <https://forestry-labs.github.io/distillML/index.html>
Brian Cho, Theo F. Saarinen, Jasjeet S. Sekhon, Simon Walter.

**License** GPL (>= 3)

**Encoding** UTF-8

**Imports** ggplot2, glmnet, Rforestry, MASS, dplyr, R6 (>= 2.0), checkmate, purrr, tidyr, data.table, mltools, gridExtra

**Suggests** testthat, knitr, rmarkdown, mvtnorm

**Collate** 'predictor.R' 'interpret.R' 'distiller.R' 'plotter.R' 'surrogate.R'

**RoxygenNote** 7.1.2

**NeedsCompilation** no

**Author** Brian Cho [aut],
Theo Saarinen [aut, cre],
Jasjeet Sekhon [aut],
Simon Walter [aut]

**Repository** CRAN

**Date/Publication** 2022-07-07 08:50:06 UTC

# R topics documented:



---

`ale` *Constructs an ALE for a model.*

---

## Description

Constructs an ALE for a model.

## Usage

```
ale(
  predict_function,
  num_grid_points,
  training_data,
  variable_names,
  center = "zero",
  grid_points,
  window_size
)
```

## Arguments

`predict_function`
: a function taking a single tibble argument and returning the model predictions corresponding to that tibble.

| | |
|---|---|
| num_grid_points | the number of grid_points at which to construct the ALE |
| training_data | the training data used to fit the model |
| variable_names | a vector of feature names in training data for which an ALE is required. |
| center | one of "uncentered" meaning the plots are not centered, "mean" meaning the plots are centered at their mean and "zero" meaning the ALE passes through the origin. When using center == "zero" we recommend setting window_size because otherwise a smaller and possibly empty set will be used to compute the ALE at zero. |
| grid_points | The grid points to use for the AlE calculation. |
| window_size | the fraction of the data (between zero and one) used to compute each ALE point. |

---

`build.grid` *Build grid used for weights in distilled surrogate model*

---

## Description

A dataframe storing the true predictions and the PDP predictions

## Usage

```
build.grid(object, feat.ind = 1:length(object$features))
```

## Arguments

| | |
|---|---|
| object | The Interpreter object |
| feat.ind | The indices of the features in the Interpreter's features that we want to include as PDP functions in the distilled model. |

## Value

A dataframe used to find weights in regression (one-hot encoding for categorical features)

## Note

This function is mainly used as a subroutine for the distill function. We include this as a public function to allow users to create their own weights and surrogate functions outside of our implemented method.

`center.preds` *Centers the predicted values for 1-d ICE and PDP plots or 2-d PDP plots*

## Description

Given the specified 'center.at' values of the Interpreter object, this function centers all of the plots in the Interpreter object of the specified type of plot.

## Usage

```
center.preds(object, features = NULL, plot.type, feats.2d = NULL)
```

## Arguments

| | |
|---|---|
| `object` | The Interpreter object to use |
| `features` | A vector of names for the 1-D features we want to center |
| `plot.type` | The type of plot that the user wants to center the predictions of. should be one of either "ICE", "PDP.1D", or "PDP.2D" |
| `feats.2d` | A 2-column dataframe or matrix that gives the first variable in in the first column, and the second variable in the next. The number of rows is equal to the number of 2-D PDPs one would like to center. |

## Details

center.preds

## Value

A list of centered data frame/matrix of values for the plot

## Note

This function is mainly used to examine the exact values in the plot if the plot is centered. Note that this function should only be called after calling one of the various predict functions that matches the 'plot.type' parameter with 'save' equal to TRUE.

distill *Builds surrogate model from an interpreter object based on the univariate PDP functions of the original model.*

## Description

Builds a surrogate model from the PDP functions

## Usage

```
distill(
  object,
  center.mean = TRUE,
  features = 1:length(object$features),
  cv = FALSE,
  snap.grid = TRUE,
  snap.train = TRUE,
  params.glmnet = list(),
  params.cv.glmnet = list()
)
```

## Arguments

| | |
|---|---|
| object | The Interpreter object |
| center.mean | Boolean value that determines whether to center each column of predictions by their respective means. Default is TRUE |
| features | The indices of the features in the Interpreter's features that we want to include as PDP functions in the distilled model. |
| cv | Boolean that indicates whether we want to cross-validate our fitted coefficients with a regularizer. This should only be done when regularizing coefficients. |
| snap.grid | Boolean function that determines whether the model recalculates each value predicted or uses an approximation from previous calculations. When this parameter is set to TRUE, we approximate the predicted values with prevoius calculations. Default is TRUE. |
| snap.train | Boolean that determines whether we use the training data or the equally spaced grid points. By default, this is true, which means we snap to grid points as determined by the training data's marginal distribution. |
| params.glmnet | Optional list of parameters to pass to glmnet while fitting PDP curves to resemble the original predictions. By specifying parameters, one can do lasso or ridge regression. |
| params.cv.glmnet | Optional list of parameters to pass to cv.glmnet while fitting PDP curves to resemble the original predictions. By specifying parameters, one can do lasso or ridge regression. |

## Value

A surrogate class object that can be used for predictions

## Note

For further details, please refer to the vignette for this method, which includes usage examples.

---

`Interpreter` *Interpreter class description*

---

## Description

A wrapper class based on a predictor object for examining the predictions of the model with respect to one or two features. The two methods for interpreting a model based on one or two features are partial dependence plots (PDP), which averages over the marginal distribution of the predictions of the model, and accumulated local effects (ALE) functions which averages over the conditional distribution of the predictions of the model.

The only necessary argument is the Predictor object. The other arguments are optional, but it may be useful to specify the number of samples or the specific data points (data.points) if the training data is very large. This can greatly reduce the time for computation.

For the output, the model returns an interpreter object with two lists of functions: one for interpreting a single feature's role in the black-box model, and the other for intepreting a pair of features' role in the black-box model. These interpretability functions are built for each possible feature (or pair of features). Each of these functions return a vector of averaged predictions equal in length to the number of values (or number of rows) input into the function.

## Public fields

`predictor` The Predictor object that contains the model that the user wants to query. This is the only parameter that is required to initialize an Interpreter object. All entries in the vector must match column names from the 'data' parameter of the Predictor object.

`features` An optional list of single features that we want to create PDP functions for.

`features.2d` A two column data frame that contains pairs of names that we want to create 2D PDP functions for. All entries in the data frame must match column names from the 'data' parameter of the Predictor object.

`data.points` A vector of indices of data points in the training data frame to be used as the observations for creating the PDP/ICE/ALE plots. When the training data is large, it can greatly reduce the required computation to pass only a downsampled subset of the training data to the pdp function construction. Alternatively, if one is only interested understanding the model predictions for a specific subgroup, the indices of the observations in the given subgroup can be passed here.

`pdp.1d` A List of functions giving single feature PDP interpretations of the model.

`pdp.2d` A List of functions giving two-feature PDP interpretations of the model

`feat.class` A vector that contains the class for each feature (categorical or continuous)

`center.at` The value(s) to center the feature plots at. A list of equal length to the length of the features.

`grid.points` A list of vectors containing the grid points to use for the predictions for PDP and ICE plots. For ALE plots, we use quantile-based methods that depend on the distribution of the training data.

`grid.size` The number of grid points to plot for a continuous feature. This parameter sets the number of grid points for PDP, ICE, and ALE plots.

`saved` A list that caches the previous calculations for the 1-D ICE plots, 1-D PDP plots, 2-D PDP plots, and grid points for building the distilled model. This saves the uncentered calculations.

`ale.grid` A list that caches the saved predictions for the ALE plots

## Methods

### Public methods:

- `Interpreter$new()`
- `Interpreter$clone()`

### Method `new()`:

*Usage:*

```
Interpreter$new(
  predictor = NULL,
  samples = 1000,
  data.points = NULL,
  grid.size = 50
)
```

*Arguments:*

`predictor` The Predictor object that contains the model that the user wants to query. This is the only parameter that is required to initialize an Interpreter object. All entries in the vector must match column names from the ‘data‘ parameter of the Predictor object.

`samples` The number of observations used for the interpretability method. If no number is given, the default set is the minimum between 1000 and the number of rows in the training data set. Rows with missing values are excluded from being sampled.

`data.points` The indices of the data points used for the PDP/ALE. This overwrites the "samples" parameter above.

`grid.size` The number of grid points used to create for the PDP, ICE, and ALE plots for each feature.

*Returns:* An ‘Interpreter‘ object.

### Method `clone()`: The objects of this class are cloneable with this method.

*Usage:*

```
Interpreter$clone(deep = FALSE)
```

*Arguments:*

`deep` Whether to make a deep clone.

## Note

The class that wraps a Predictor object for application of different interpretability methods. For usage examples, please refer to the README document.

## Examples

```
library(distillML)
library(Rforestry)
set.seed(491)
data <- MASS::crabs

levels(data$sex) <- list(Male = "M", Female = "F")
levels(data$sp) <- list(Orange = "O", Blue = "B")
colnames(data) <- c("Species","Sex","Index","Frontal Lobe",
"Rear Width", "Carapace Length","Carapace Width","Body Depth")

test_ind <- sample(1:nrow(data), nrow(data)%/%5)
train_reg <- data[-test_ind,]
test_reg <- data[test_ind,]


forest <- forestry(x=train_reg[,-which(names(train_reg)=="Carapace Width")],
y=train_reg[,which(names(train_reg)=="Carapace Width")])

forest_predictor <- Predictor$new(model = forest, data=train_reg,
y="Carapace Width", task = "regression")

forest_interpret <- Interpreter$new(predictor = forest_predictor)
```

---

# localSurrogate *Given a interpreter object with at least one pair of features, create a small surrogate model for the two features using the PDP function as the output and the two features as the independent variables.*

---

## Description

Plots and returns a Rforestry object with a single tree explaining the PDP surface.

## Usage

```
localSurrogate(
  object,
  features.2d = NULL,
  interact = FALSE,
  params.forestry = list()
)
```

### Arguments

| | |
|---|---|
| `object` | Interpreter object to make plots + surrogate for. |
| `features.2d` | A two-column dataframe of pairs of features to make local surrogates for. Each row represents a pair of features, with the names of features as the entries. |
| `interact` | An indicator specifying if the surrogate model should also be given the interaction terms to create the surrogate models with. Default is FALSE. |
| `params.forestry` | Optional list of parameters to pass to the surrogate model. Defaults to the standard Rforestry parameters with ntree = 1 and maxDepth = 2. |

### Value

A list of two distinct lists: one list contains the local surrogate models, and the other containing the 2-D PDP plots for the specified features.

---

## `plot-Interpreter` *Plotting method for Interpretor model*

---

### Description

Plots the PDP, ALE, or ICE plots for an Interpreter object

### Usage

```
## S3 method for class 'Interpreter'
plot(
  x,
  method = "pdp+ice",
  features = NULL,
  features.2d = NULL,
  clusters = NULL,
  clusterType = "preds",
  smooth = FALSE,
  smooth.bandwidth = NULL,
  smooth.kernel = "normal",
  smooth.npoints = 2 * x$grid.size,
  ...
)
```

### Arguments

| | |
|---|---|
| `x` | Interpreter object to generate plots from |
| `method` | The type of plot that we want to generate. Must be one of "ice", "pdp+ice", "pdp", or "ale" |
| `features` | a vector of feature names that we want to produce 1-D plots for. |

| | |
|---|---|
| features.2d | 2-D features that we want to produce plots for arguments. A two-column dataframe of pairs of features to make local surrogates for. Each row represents a pair of features, with the names of features as the entries.If the 'method' parameter is set to "ale", this argument should not be used. |
| clusters | A number of clusters to cluster the ICE predictions with. If this is not NULL, one must use the method "ice". |
| clusterType | An indicator specifying what method to use for the clustering. The possible options are "preds", and "gradient". If "preds" is used, the clusters will be determined by running K means on the predictions of the ICE functions. If the "gradient" option is used, the clusters will be determined by running K means on the numerical gradient of the predictions of the ICE functions. If this is not NULL, one must use the method "ice". |
| smooth | A binary variable to determine whether to smoothen the plots of the PDP, ICE, or ALE curves for continuous variables. |
| smooth.bandwidth | The bandwidth for the kernels. They are scaled such that their quartiles are at 0.25 * bandwidth. By default, this is set as the maximum difference between the minimum and maximum of the grid points. |
| smooth.kernel | The type of kernel to be used. Users can input either strings "box" or "normal". The default is "normal". |
| smooth.npoints | The number of points returned when using the kernel method. By default, this is twice the number of grid points for that feature. |
| ... | Additiional parameters to pass to the plot function |

## Value

A list of plots with 1-d features and 2-d features. For 2-d features with one continuous and one categorical feature, the plot is a linear plot of the continuous feature with group colors representing the categorical feature. For two continuous features, the plot is a heatmap with the shade representing the value of the outcome.

---

# predict-Predictor *Predict method for Predictor class*

---

## Description

Gives a single column of predictions from a model that is wrapped by the Predictor object

## Usage

```
## S3 method for class 'Predictor'
predict(object, newdata, ...)
```

### Arguments

| | |
|---|---|
| object | The Predictor object to use to make predictions. |
| newdata | The data frame to use for the independent features in the prediction. |
| ... | Additional arguments that are passed to the model predict function. For instance, these can be different aggregation options (aggregation = "oob") that are accepted by the prediction function of the model. |

### Value

A data frame with a single column containing the predictions for each row of the newdata data frame.

---

## predict-Surrogate *Prediction method for the distilled surrogate model*

---

### Description

Predicts outputs given new data

### Usage

```
## S3 method for class 'Surrogate'
predict(object, newdata, ...)
```

### Arguments

| | |
|---|---|
| object | A surrogate object distilled from the interpreter |
| newdata | The dataframe to use for the predictions |
| ... | Additional parameters to pass to predict |

### Value

A one-column dataframe of the surrogate model's predictions

`Predictor` *Predictor class description*

## Description

A wrapper class for generic ML algorithms (xgboost, RF, BART, rpart, etc.) in order to standardize the predictions given by different algorithms to be compatible with the interpretability functions.

The necessary variables are model, data, y. The other variables are optional, and depend on the use cases. Type should be used only when a prediction function is NOT specified.

The outputs of the algorithm must be the values if it is regression, or probabilities if classification. For classification problems with more than two categories, the output comes out as vectors of probabilities for the specified "class" category. Because this is for ML interpretability, other types of predictions (ex: predictions that spit out the factor) are not allowed.

## Public fields

`data` The training data that was used during training for the model. This should be a data frame matching the data frame the model was given for training, which includes the label or outcome.

`model` The object corresponding to the trained model that we want to make a Predictor object for. If this model doesn't have a generic predict method, the user has to provide a custom predict function that accepts a data frame.

`task` The prediction task the model is trained to perform (‘classification‘ or ‘regression‘).

`class` The class for which we get predictions. We specify this to get the predictions (such as probabilites) for an observation being in a specific class (e.g. Male or Female). This parameter is necessary for classification predictions with more than a single vector of predictions.

`prediction.function` An optional parameter if the model doesn't have a generic prediction function. This should take a data frame and return a vector of predictions for each observation in the data frame.

`y` The name of the outcome feature in the ‘data‘ data frame.

## Methods

### Public methods:

- `Predictor$new()`
- `Predictor$clone()`

### Method `new()`:

*Usage:*

```
Predictor$new(
  model = NULL,
  data = NULL,
  predict.func = NULL,
  y = NULL,
  task = NULL,
```

```
  class = NULL,
  type = NULL
)
```

*Arguments:*

`model` The object corresponding to the trained model that we want to make a Predictor object for. If this model doesn't have a generic predict method, the user has to provide a custom predict function that accepts a data frame.

`data` The training data that was used during training for the model. This should be a data frame matching the data frame the model was given for training, including the label or outcome.

`predict.func` An optional parameter if the model doesn't have a generic prediction function. This should take a data frame and return a vector of predictions for each observation in the data frame.

`y` The name of the outcome feature in the 'data' data frame.

`task` The prediction task the model is trained to perform ('classification' or 'regression').

`class` The class for which we get predictions. We specify this to get the predictions (such as probabilites) for an observation being in a specific class (e.g. Male or Female). This parameter is necessary for classification predictions with more than a single vector of predictions.

`type` The type of predictions done (i.e. 'response' for predicted probabliities for classification). This feature should only be used if no predict.func is specified.

*Returns:* A 'Predictor' object.

**Method** `clone()`**:** The objects of this class are cloneable with this method.

*Usage:*

```
Predictor$clone(deep = FALSE)
```

*Arguments:*

`deep` Whether to make a deep clone.

## Note

The class that wraps a machine learning model in order to provide a standardized method for predictions for different models. prediction method must be constructed, with optional argument of type

## Examples

```
library(distillML)
library(Rforestry)
set.seed(491)
data <- MASS::crabs

levels(data$sex) <- list(Male = "M", Female = "F")
levels(data$sp) <- list(Orange = "O", Blue = "B")
colnames(data) <- c("Species","Sex","Index","Frontal Lobe",
"Rear Width", "Carapace Length","Carapace Width","Body Depth")

test_ind <- sample(1:nrow(data), nrow(data)%/%5)
train_reg <- data[-test_ind,]
```

```
test_reg <- data[test_ind,]


forest <- forestry(x=train_reg[,-which(names(train_reg)=="Carapace Width")],
y=train_reg[,which(names(train_reg)=="Carapace Width")])

forest_predictor <- Predictor$new(model = forest, data=train_reg,
y="Carapace Width", task = "regression")
```

---

`predict_ALE` *Prediction function for the ALE plots*

---

## Description

Prediction function for the ALE plots

## Usage

```
predict_ALE(x, feature, training_data, save = TRUE)
```

## Arguments

| | |
|---|---|
| `x` | An interpreter object |
| `feature` | The feature to build the ALE for (must be continuous) |
| `training_data` | The training data to use in order to build the ALE |
| `save` | Boolean to save the ALE predictions |

## Value

A tibble that contains the ALE predictions for the given values

---

`predict_ICE.Plotter` *Prediction Function for ICE Plots*

---

## Description

Gives predictions at each point on the grid.

## Usage

```
predict_ICE.Plotter(object, features = NULL, save = TRUE)
```

### Arguments

| | |
|---|---|
| object | The Interpeter object to use. |
| features | A vector with the names of the features to predict ICE plots for |
| save | A boolean indicator to indicate whether the calculations should be saved in the interpreter object or not. This can help reduce computation if the ICE functions are used many times, but requires additional memory to store the predictions. By default, this is TRUE. |

### Value

A list of data frames, one for each feature. In each data frame, the first column contains the grid values for the feature, and each subsequent column has a single observation corresponding to the prediction of the model when with the given feature set to that grid point value.

### Note

This method is meant to primarily be used to find the exact values for the ICE curves plotted. Note that after the PDP curve is plotted, the returned object of this function will be the saved predictions for plotting the curve, rather than a recalculation of the values.

---

## predict_PDP.1D.Plotter

*Prediction Function for PDP Plots*

---

### Description

Gives prediction curve for all specified features in the plotter object

### Usage

```
predict_PDP.1D.Plotter(object, features = NULL, save = TRUE)
```

### Arguments

| | |
|---|---|
| object | The Interpreter object to plot PDP curves for. |
| features | A vector with the names of the features to predict ICE plots for |
| save | A boolean indicator to indicate whether the calculations should be saved in the interpreter object or not. This can help reduce computation if the PDP functions are used many times, but requires additional memory to store the predictions. By default, this is set to TRUE. |

### Details

predict_PDP.1D.Plotter

### Value

A list of data frames with the grid points and PDP prediction values for each feature in object

### Note

This method is meant to primarily be used to find the exact values for the 1-D PDP curves plotted. Note that after the PDP curve is plotted, the returned object of this function will be the saved predictions for plotting the curve, rather than a recalculation of the values.

---

## predict_PDP.2D.Plotter

*Two Dimensional Prediction Curve for PDP Plots*

---

### Description

Gives prediction surface for all specified feature pairs in the interpreter object (features.2d)

### Usage

```
predict_PDP.2D.Plotter(object, feat.2d, save = TRUE)
```

### Arguments

| | |
|---|---|
| `object` | The Interpreter object to use. |
| `feat.2d` | A 2-column dataframe or matrix that gives the first variable in in the first column, and the second variable in the next. The number of rows is equal to the number of 2-D PDPs one would like. |
| `save` | A boolean indicator to indicate whether the calculations should be saved in the interpreter object or not. This can help reduce computation if the PDP functions are used many times, but requires additional memory to store the predictions. By default, this is set to TRUE. |

### Details

predict_PDP.2D.Plotter

### Value

A list of data frames for each pair of features.2d. Each data frame contains columns corresponding to the grid points for the two selected features and a column corresponding to the predictions of the model at the given combination of grid points.

### Note

This method is meant to primarily be used to find the exact values for the 2-D PDP curves or heatmap plotted. Note that after the PDP curve is plotted, the returned object of this function will be the saved predictions for plotting the curve, rather than a recalculation of the values.

## print-Predictor *The Printing method for Predictor class*

### Description

Prints the task of an instance of the Predictor class.

### Usage

```
## S3 method for class 'Predictor'
print(x, ...)
```

### Arguments

| | |
|---|---|
| x | The Predictor object to print |
| ... | Additional arguments passed to the print function. |

## set.center.at *Sets a new center in the PDP and ICE plots made by an Interpreter*

### Description

Method for setting center value for a specific feature

### Usage

```
set.center.at(object, feature, value)
```

### Arguments

| | |
|---|---|
| object | The Interpreter class that we want to recenter the plots of. |
| feature | The name of the feature to set grid points for. |
| value | The new value to use for the plots of the specified feature to be centered at. Must match the type of the feature (a factor level or continuous value in the range of the specified feature). |

### Note

Unlike the grid predictions, the center.at values do not modify any of the previous saved calculations. Therefore, it does not change or remove any of the previously calculated, saved data. These center values are simply for the plots made by the interpreter object, rather than the distilled model.

## set.grid.points *Sets grid points used for plotting PDP and ICE plots*

### Description

Method for setting grid points for a specific feature plot

### Usage

```
set.grid.points(object, feature, values)
```

### Arguments

| | |
|---|---|
| object | The Interpreter class that we want to modify the grid points of. |
| feature | The name of the feature to set grid points for. |
| values | The set of new values to be used as the grid points for the selected feature. Must be a vector with entries in the range of the feature values in the training set and must match the type of the given feature (either a vector of factor levels or a vector of continuous feature values). Note that the center must be within the range of new grid points for continuous features. |

### Note

Because the grid points determine what calculations are performed for the PDP/ICE functions, changing the grid points will remove any of the previously calculated values in the 'Interpreter' object. For any 1-D ICE or PDP plot, it will remove the previous calculations for the given feature. For any 2-D PDP calcuations, it will remove plots that include the given feature as any of its features. Note that these set grid points only apply to PDP and ICE plots, and ALE plots have their own grid points determined by the distribution of the training data.

## Surrogate *Surrogate class description*

### Description

The class for distilled surrogate models.

### Public fields

`interpreter` The interpreter object to use as a standardized wrapper for the model

`features` The indices of the features in the data used in the surrogate model

`weights` The weights used to recombine the PDPs into a surrogate for the original model

`intercept` The intercept term we use for our predictions

`feature.centers` The center value for the features determined in the model

`center.mean` Boolean value that determines whether we use the mean-centered data for our predictions

`grid` A list of PDPS that determine our prediction.

`snap.grid` Boolean that determines whether we use grid.points

## Methods

### Public methods:

- `Surrogate$new()`
- `Surrogate$clone()`

### Method `new()`:

*Usage:*

```
Surrogate$new(
  interpreter,
  features,
  weights,
  intercept,
  feature.centers,
  center.mean,
  grid,
  snap.grid
)
```

*Arguments:*

`interpreter` The interpreter object we want to build a surrogate model for.

`features` The indices of features in the training data used for the surrogate model

`weights` The weights for each given feature after the surrogate model is fit.

`intercept` The baseline value. If uncentered, this is 0, and if centered, this will be the mean of the predictions of the original model on the training data.

`feature.centers` The baseline value for the effect of each feature. If uncentered, this is 0.

`center.mean` A boolean value that shows whether this model is a centered or uncentered model

`grid` A list of dataframes containing the pre-calculated values used to generate predictions if snap.grid is TRUE

`snap.grid` Boolean that determines if we use previously calculated values or re-predict using the functions.

*Returns:* A surrogate model object that we can use for predictions

### Method `clone()`: The objects of this class are cloneable with this method.

*Usage:*

```
Surrogate$clone(deep = FALSE)
```

*Arguments:*

`deep` Whether to make a deep clone.

## Note

Do not initalize this class on its own. It is automatically created by the distill function for the interpreter class.

## Examples

```
library(distillML)
library(Rforestry)
set.seed(491)
data <- MASS::crabs

levels(data$sex) <- list(Male = "M", Female = "F")
levels(data$sp) <- list(Orange = "O", Blue = "B")
colnames(data) <- c("Species","Sex","Index","Frontal Lobe",
"Rear Width", "Carapace Length","Carapace Width","Body Depth")

test_ind <- sample(1:nrow(data), 180)
train_reg <- data[-test_ind,]
test_reg <- data[test_ind,]


forest <- forestry(x=train_reg[,-which(names(train_reg)=="Carapace Width")],
y=train_reg[,which(names(train_reg)=="Carapace Width")])

forest_predictor <- Predictor$new(model = forest, data=train_reg,
y="Carapace Width", task = "regression")

forest_interpret <- Interpreter$new(predictor = forest_predictor)

# Both initializations of a surrogate class result in the same surrogate model
surrogate.model <- distill(forest_interpret)
surrogate.model <- distill(forest_interpret,
                           center.mean = TRUE,
                           features = 1:length(forest_interpret$features),
                           cv = FALSE,
                           snap.grid = TRUE,
                           snap.train = TRUE)
```

# Index