Package 'autokeras'

February 20, 2020

Type Package

Title R Interface to 'AutoKeras'

Version 1.0.1

Maintainer Juan Cruz Rodriguez <jcrodriguez@unc.edu.ar>

Description R Interface to 'AutoKeras' <https://autokeras.com/>.
'AutoKeras' is an open source software library for Automated Machine Learning (AutoML). The ultimate goal of AutoML is to provide easily accessible deep learning tools to domain experts with limited data science or machine learning background. 'AutoKeras' provides functions to automatically search for architecture and hyperparameters of deep learning models.

Encoding UTF-8

License MIT + file LICENSE

URL https://github.com/r-tensorflow/autokeras

BugReports https://github.com/r-tensorflow/autokeras/issues

LazyData true

Depends R (>= 3.1)

Imports keras, methods, reticulate, stats

RoxygenNote 7.0.2

Suggests testthat, covr

NeedsCompilation no

Author Juan Cruz Rodriguez [aut, cre],
Data Analytics at Texas A&M (DATA) Lab [cph]

Repository CRAN

Date/Publication 2020-02-20 09:50:03 UTC

R topics documented:

`autokeras-package` *R Interface to AutoKeras*

Description

[AutoKeras](https://autokeras.com/) is an open source software library for automated machine learning (AutoML). It is developed by [DATA Lab](http://faculty.cs.tamu.edu/xiahu/index.html) at Texas A&M University and community contributors. The ultimate goal of AutoML is to provide easily accessible deep learning tools to domain experts with limited data science or machine learning background. AutoKeras provides functions to automatically search for architecture and hyperparameters of deep learning models.

Author(s)

Maintainer: Juan Cruz Rodriguez <jcrodriguez@unc.edu.ar>

Other contributors:

- Data Analytics at Texas A&M (DATA) Lab [copyright holder]

See Also

Useful links:

- https://github.com/r-tensorflow/autokeras
- Report bugs at https://github.com/r-tensorflow/autokeras/issues

AutokerasModel-class *Autokeras Model Class Representation*

Description

Autokeras Model Class Representation

evaluate *Evaluate a Model*

Description

Evaluate the best model for the given data.

Usage

```
## S3 method for class 'AutokerasModel'
evaluate(object, x_test, y_test = NULL, batch_size = 32, ...)
```

Arguments

object	: A trained AutokerasModel instance.
x_test	: Any allowed types according to the input node. Testing data. Check corresponding AutokerasModel help to note how it should be provided.
y_test	: Any allowed types according to the input node. Testing data. Check corresponding AutokerasModel help to note how it should be provided. Defaults to 'NULL'.
batch_size	: numeric. Defaults to '32'.
...	: Unused.

Value

numeric test loss (if the model has a single output and no metrics) or list of scalars (if the model has multiple outputs and/or metrics). The attribute model$metrics_names will give you the display labels for the scalar outputs.

Examples

```
## Not run:
library("keras")

# use the MNIST dataset as an example
mnist <- dataset_mnist()
c(x_train, y_train) %<-% mnist$train
c(x_test, y_test) %<-% mnist$test
```

```
library("autokeras")

# Initialize the image classifier
clf <- model_image_classifier(max_trials = 10) %>% # It tries 10 different models
  fit(x_train, y_train) # Feed the image classifier with training data

# Predict with the best model
(predicted_y <- clf %>% predict(x_test))

# Evaluate the best model with testing data
clf %>% evaluate(x_test, y_test)

# Get the best trained Keras model, to work with the keras R library
export_model(clf)

## End(Not run)
```

`export_model` *Export Model*

Description

Export the best trained Keras Model.
Actually, exporting the model as a Keras model is not working as expected, check out the bug https://github.com/keras-team/autokeras/issues/929 .

Usage

```
export_model(autokeras_model)
```

Arguments

`autokeras_model` : A trained AutokerasModel instance.

Value

keras.Model instance. The best model found during the search, loaded with trained weights.

Examples

```
## Not run:
library("keras")

# use the MNIST dataset as an example
mnist <- dataset_mnist()
c(x_train, y_train) %<-% mnist$train
c(x_test, y_test) %<-% mnist$test
```

```
library("autokeras")

# Initialize the image classifier
clf <- model_image_classifier(max_trials = 10) %>% # It tries 10 different models
  fit(x_train, y_train) # Feed the image classifier with training data

# Predict with the best model
(predicted_y <- clf %>% predict(x_test))

# Evaluate the best model with testing data
clf %>% evaluate(x_test, y_test)

# Get the best trained Keras model, to work with the keras R library
export_model(clf)

## End(Not run)
```

`fit` *Search for the Best Model and Hyperparameters*

Description

It will search for the best model and hyperparameters based on the performances on validation data.

Usage

```
## S3 method for class 'AutokerasModel'
fit(
  object,
  x = NULL,
  y = NULL,
  epochs = 1000,
  callbacks = NULL,
  validation_split = 0.2,
  validation_data = NULL,
  ...
)
```

Arguments

`object`	: An AutokerasModel instance.
`x`	: Training data x. Check corresponding AutokerasModel help to note how it should be provided.
`y`	: Training data y. Check corresponding AutokerasModel help to note how it should be provided.

epochs : numeric. The number of epochs to train each model during the search. If unspecified, by default we train for a maximum of ‘1000‘ epochs, but we stop training if the validation loss stops improving for 10 epochs (unless you specified an EarlyStopping callback as part of the ‘callbacks‘ argument, in which case the EarlyStopping callback you specified will determine early stopping).

callbacks : list of Keras callbacks to apply during training and validation.

validation_split
: numeric between 0 and 1. Defaults to ‘0.2‘. Fraction of the training data to be used as validation data. The model will set apart this fraction of the training data, will not train on it, and will evaluate the loss and any model metrics on this data at the end of each epoch. The validation data is selected from the last samples in the ‘x‘ and ‘y‘ data provided, before shuffling. This argument is not supported when ‘x‘ is a dataset. The best model found would be fit on the entire dataset including the validation data.

validation_data
: Data on which to evaluate the loss and any model metrics at the end of each epoch. The model will not be trained on this data. ‘validation_data‘ will override ‘validation_split‘. The type of the validation data should be the same as the training data. The best model found would be fit on the training dataset without the validation data.

... : Unused.

Value

A trained AutokerasModel.

Examples

```
## Not run:
library("keras")

# use the MNIST dataset as an example
mnist <- dataset_mnist()
c(x_train, y_train) %<-% mnist$train
c(x_test, y_test) %<-% mnist$test

library("autokeras")

# Initialize the image classifier
clf <- model_image_classifier(max_trials = 10) %>% # It tries 10 different models
  fit(x_train, y_train) # Feed the image classifier with training data

# If you want to use own valitadion data do:
clf <- model_image_classifier(max_trials = 10) %>%
  fit(
    x_train,
    y_train,
    validation_data = list(x_test, y_test)
  )
```

```
# Predict with the best model
(predicted_y <- clf %>% predict(x_test))

# Evaluate the best model with testing data
clf %>% evaluate(x_test, y_test)

# Get the best trained Keras model, to work with the keras R library
export_model(clf)

## End(Not run)
```

`install_autokeras` *Install Autokeras, Keras, and the Tensorflow Backend*

Description

AutoKeras, Keras, and TensorFlow will be installed into an "r-tensorflow" virtual or conda environment. Note that "virtualenv" is not available on Windows (as this isn't supported by TensorFlow).

Usage

```
install_autokeras(
  method = c("auto", "virtualenv", "conda"),
  conda = "auto",
  version = "1.0.1",
  keras = "default",
  tensorflow = "2.1.0",
  extra_packages = NULL,
  ...
)
```

Arguments

`method`	Installation method ("virtualenv" or "conda")
`conda`	Path to conda executable (or "auto" to find conda using the PATH and other conventional install locations).
`version`	Version of AutoKeras to install. Specify "default" to install the latest release. Otherwise specify an alternate version (e.g. "0.3.5"). The default value is "1.0.1" as it is the latest tested version.
`keras`	Keras version to install. Specify "default" to install the latest release. Otherwise specify an alternate version (e.g. "2.2.2").
`tensorflow`	TensorFlow version to install. Specify "default" to install the CPU version of the latest release. Specify "gpu" to install the GPU version of the latest release. You can also provide a full major.minor.patch specification (e.g. "1.1.0"), appending "-gpu" if you want the GPU version (e.g. "1.1.0-gpu"). Alternatively, you can provide the full URL to an installer binary (e.g. for a nightly binary).

extra_packages	Additional PyPI packages to install along with Keras and TensorFlow.
...	Other arguments passed to tensorflow::install_tensorflow().

Value

None

GPU Installation

Keras and TensorFlow can be configured to run on either CPUs or GPUs. The CPU version is much easier to install and configure so is the best starting place especially when you are first learning how to use Keras. Here's the guidance on CPU vs. GPU versions from the TensorFlow website:

- *TensorFlow with CPU support only*. If your system does not have a NVIDIA® GPU, you must install this version. Note that this version of TensorFlow is typically much easier to install, so even if you have an NVIDIA GPU, we recommend installing this version first.

- *TensorFlow with GPU support*. TensorFlow programs typically run significantly faster on a GPU than on a CPU. Therefore, if your system has a NVIDIA® GPU meeting all prerequisites and you need to run performance-critical applications, you should ultimately install this version.

To install the GPU version:

1) Ensure that you have met all installation prerequisites including installation of the CUDA and cuDNN libraries as described in [TensorFlow GPU Prerequistes](https://tensorflow.rstudio.com/installation_gpu.html#prereq

2) Pass ‘tensorflow = "gpu"‘ to ‘install_autokeras()‘. For example:

“‘ install_autokeras(tensorflow="gpu") ““

Windows Installation

The only supported installation method on Windows is "conda". This means that you should install Anaconda 3.x for Windows prior to installing Keras.

Custom Installation

Installing Keras and TensorFlow using ‘install_autokeras()‘ isn't required to use the Keras R package. You can do a custom installation of Keras (and desired backend) as described on the [Keras website](https://keras.io/#installation) and the Keras R package will find and use that version.

See the documentation on [custom installations](https://tensorflow.rstudio.com/installation.html#custom-installation) for additional information on how version of Keras and TensorFlow are located by the Keras package.

Additional Packages

If you wish to add additional PyPI packages to your Keras / TensorFlow environment you can either specify the packages in the ‘extra_packages‘ argument of ‘install_autokeras()‘, or alternatively install them into an existing environment using the [reticulate::py_install()] function.

Examples

```
## Not run:

# default installation
library("autokeras")
install_autokeras()

# install using a conda environment (default is virtualenv)
install_autokeras(method = "conda")

# install with GPU version of TensorFlow
# (NOTE: only do this if you have an NVIDIA GPU + CUDA!)
install_autokeras(tensorflow = "gpu")

# install a specific version of TensorFlow
install_autokeras(tensorflow = "1.2.1")
install_autokeras(tensorflow = "1.2.1-gpu")

# install a specific version of Keras and TensorFlow
install_autokeras(keras = "2.2.2", tensorflow = "1.2.1")

## End(Not run)
```

`load_model` *Load Model*

Description

Load an AutoKeras Model.

Usage

```
load_model(filename)
```

Arguments

`filename`	: A character string naming a file to save the model.

Value

A previously saved AutokerasModel.

Examples

```
## Not run:
library("keras")

# use the MNIST dataset as an example
```

```
mnist <- dataset_mnist()
c(x_train, y_train) %<-% mnist$train
c(x_test, y_test) %<-% mnist$test

library("autokeras")

# Initialize the image classifier
clf <- model_image_classifier(
  directory = ".",
  name = "autokeras_mnist",
  max_trials = 10
) %>% # It tries 10 different models
  fit(x_train, y_train) # Feed the image classifier with training data

# Predict with the best model
(predicted_y <- clf %>% predict(x_test))

# Evaluate the best model with testing data
clf %>% evaluate(x_test, y_test)

# Get the best trained Keras model, to work with the keras R library
export_model(clf)

# Save the AutoKeras model.
# Make sure that `directory` and `name` were set when creating the model.
save_model(clf, "my_model.pkl")

# And load it again
new_clf <- load_model("my_model.pkl")

## End(Not run)
```

`model_image_classifier` *AutoKeras Image Classifier Model*

Description

AutoKeras image classification class.
It is used for image classification. It searches convolutional neural network architectures for the best configuration for the image dataset. To ‘fit‘, ‘evaluate‘ or ‘predict‘, format inputs as:

- x : array. The shape of the data should be 3 or 4 dimensional, the last dimension of which should be channel dimension.
- y : array. It can be raw labels, one-hot encoded if more than two classes, or binary encoded for binary classification.

Usage

```
model_image_classifier(
  num_classes = NULL,
  multi_label = FALSE,
  loss = NULL,
  metrics = list("accuracy"),
  name = "image_classifier",
  max_trials = 100,
  directory = tempdir(),
  objective = "val_loss",
  overwrite = TRUE,
  seed = runif(1, 0, 1e+07)
)
```

Arguments

num_classes	: numeric. Defaults to 'NULL'. If 'NULL', it will infer from the data.
multi_label	: logical. Defaults to 'FALSE'.
loss	: A Keras loss function. Defaults to use 'binary_crossentropy' or 'categorical_crossentropy' based on the number of classes.
metrics	: A list of Keras metrics. Defaults to use 'accuracy'.
name	: character. The name of the AutoModel. Defaults to "image_classifier".
max_trials	: numeric. The maximum number of different Keras Models to try. The search may finish before reaching the 'max_trials'. Defaults to '100'.
directory	: character. The path to a directory for storing the search outputs. Defaults to 'tempdir()', which would create a folder with the name of the AutoModel in the current directory.
objective	: character. Name of model metric to minimize or maximize, e.g. "val_accuracy". Defaults to "val_loss".
overwrite	: logical. Defaults to 'TRUE'. If 'FALSE', reloads an existing project of the same name if one is found. Otherwise, overwrites the project.
seed	: numeric. Random seed. Defaults to 'runif(1, 0, 10e6)'.

Details

Important: The object returned by this function behaves like an R6 object, i.e., within function calls with this object as parameter, it is most likely that the object will be modified. Therefore it is not necessary to assign the result of the functions to the same object.

Value

A non-trained image classifier AutokerasModel.

Examples

```
## Not run:
library("keras")

# use the MNIST dataset as an example
mnist <- dataset_mnist()
c(x_train, y_train) %<-% mnist$train
c(x_test, y_test) %<-% mnist$test

library("autokeras")

# Initialize the image classifier
clf <- model_image_classifier(max_trials = 10) %>% # It tries 10 different models
  fit(x_train, y_train) # Feed the image classifier with training data

# If you want to use own valitadion data do:
clf <- model_image_classifier(max_trials = 10) %>%
  fit(
    x_train,
    y_train,
    validation_data = list(x_test, y_test)
  )

# Predict with the best model
(predicted_y <- clf %>% predict(x_test))

# Evaluate the best model with testing data
clf %>% evaluate(x_test, y_test)

# Get the best trained Keras model, to work with the keras R library
export_model(clf)

## End(Not run)
```

---

model_image_regressor *AutoKeras Image Regressor Model*

Description

AutoKeras image regression class.
It is used for image regression. It searches convolutional neural network architectures for the best configuration for the image dataset. To ‘fit‘, ‘evaluate‘ or ‘predict‘, format inputs as:

- x : array. The shape of the data should be 3 or 4 dimensional, the last dimension of which should be channel dimension.
- y : array. The targets passing to the head would have to be array or data.frame. It can be single-column or multi-column. The values should all be numerical.

Usage

```
model_image_regressor(
  output_dim = NULL,
  loss = "mean_squared_error",
  metrics = NULL,
  name = "image_regressor",
  max_trials = 100,
  directory = tempdir(),
  objective = "val_loss",
  overwrite = TRUE,
  seed = runif(1, 0, 1e+07)
)
```

Arguments

output_dim	: numeric. The number of output dimensions. Defaults to 'NULL'. If 'NULL', it will infer from the data.
loss	: A Keras loss function. Defaults to use "mean_squared_error".
metrics	: A list of Keras metrics. Defaults to use "mean_squared_error".
name	: character. The name of the AutoModel. Defaults to "image_regressor".
max_trials	: numeric. The maximum number of different Keras Models to try. The search may finish before reaching the 'max_trials'. Defaults to '100'.
directory	: character. The path to a directory for storing the search outputs. Defaults to 'tempdir()', which would create a folder with the name of the AutoModel in the current directory.
objective	: character. Name of model metric to minimize or maximize, e.g. "val_accuracy". Defaults to "val_loss".
overwrite	: logical. Defaults to 'TRUE'. If 'FALSE', reloads an existing project of the same name if one is found. Otherwise, overwrites the project.
seed	: numeric. Random seed. Defaults to 'runif(1, 0, 10e6)'.

Details

Important: The object returned by this function behaves like an R6 object, i.e., within function calls with this object as parameter, it is most likely that the object will be modified. Therefore it is not necessary to assign the result of the functions to the same object.

Value

A non-trained image regressor AutokerasModel.

Examples

```
## Not run:
library("keras")

# use the MNIST dataset as an example
```

```
mnist <- dataset_mnist()
c(x_train, y_train) %<-% mnist$train
c(x_test, y_test) %<-% mnist$test

library("autokeras")

# Initialize the image regressor
reg <- model_image_regressor(max_trials = 10) %>% # It tries 10 different models
  fit(x_train, y_train) # Feed the image regressor with training data

# If you want to use own valitadion data do:
reg <- model_image_regressor(max_trials = 10) %>%
  fit(
    x_train,
    y_train,
    validation_data = list(x_test, y_test)
  )

# Predict with the best model
(predicted_y <- reg %>% predict(x_test))

# Evaluate the best model with testing data
reg %>% evaluate(x_test, y_test)

# Get the best trained Keras model, to work with the keras R library
export_model(reg)

## End(Not run)
```

`model_structured_data_classifier` *AutoKeras Structured Data Classifier Model*

Description

AutoKeras structured data classification class.
To ‘fit‘, ‘evaluate‘ or ‘predict‘, format inputs as:

- x : character or array. If the data is from a csv file, it should be a character specifying the path of the csv file of the training data.
- y : character or array. If the data is from a csv file, it should be a character, which is the name of the target column. Otherwise, It can be raw labels, one-hot encoded if more than two classes, or binary encoded for binary classification.

Usage

```
model_structured_data_classifier(
  column_names = NULL,
```

```
  column_types = NULL,
  num_classes = NULL,
  multi_label = FALSE,
  loss = NULL,
  metrics = NULL,
  name = "structured_data_classifier",
  max_trials = 100,
  directory = tempdir(),
  objective = "val_accuracy",
  overwrite = TRUE,
  seed = runif(1, 0, 1e+07)
)
```

Arguments

column_names	: A list of characters specifying the names of the columns. The length of the list should be equal to the number of columns of the data excluding the target column. Defaults to 'NULL'. If 'NULL', it will obtained from the header of the csv file or the 'data.frame'.
column_types	: A list of characters. The names are the column names. The values should either be 'numerical' or 'categorical', indicating the type of that column. Defaults to 'NULL'. If not 'NULL', the 'column_names' need to be specified. If 'NULL', it will be inferred from the data.
num_classes	: numeric. Defaults to 'NULL'. If 'NULL', it will infer from the data.
multi_label	: logical. Defaults to 'FALSE'.
loss	: A Keras loss function. Defaults to use "binary_crossentropy" or "categorical_crossentropy" based on the number of classes.
metrics	: A list of Keras metrics. Defaults to use "accuracy".
name	: character. The name of the AutoModel. Defaults to "structured_data_classifier".
max_trials	: numeric. The maximum number of different Keras Models to try. The search may finish before reaching the 'max_trials'. Defaults to '100'.
directory	: character. The path to a directory for storing the search outputs. Defaults to 'tempdir()', which would create a folder with the name of the AutoModel in the current directory.
objective	: character. Name of model metric to minimize or maximize. Defaults to "val_accuracy".
overwrite	: logical. Defaults to 'TRUE'. If 'FALSE', reloads an existing project of the same name if one is found. Otherwise, overwrites the project.
seed	: numeric. Random seed. Defaults to 'runif(1, 0, 10e6)'.

Details

Important: The object returned by this function behaves like an R6 object, i.e., within function calls with this object as parameter, it is most likely that the object will be modified. Therefore it is not necessary to assign the result of the functions to the same object.

Value

A non-trained structured data classifier AutokerasModel.

Examples

```
## Not run:
library("keras")

# use the iris dataset as an example
set.seed(8818)
# balanced sample 80% for training
train_idxs <- unlist(by(seq_len(nrow(iris)), iris$Species, function(x) {
  sample(x, length(x) * .8)
}))
train_data <- iris[train_idxs, ]
test_data <- iris[-train_idxs, ]

colnames(iris)
# Species will be the interest column to predict

train_file <- paste0(tempdir(), "/iris_train.csv")
write.csv(train_data, train_file, row.names = FALSE)

# file to predict, cant have the response "Species" column
test_file_to_predict <- paste0(tempdir(), "/iris_test_2_pred.csv")
write.csv(test_data[, -5], test_file_to_predict, row.names = FALSE)

test_file_to_eval <- paste0(tempdir(), "/iris_test_2_eval.csv")
write.csv(test_data, test_file_to_eval, row.names = FALSE)

library("autokeras")

# Initialize the structured data classifier
clf <- model_structured_data_classifier(max_trials = 10) %>% # It tries 10 different models
  fit(train_file, "Species") # Feed the structured data classifier with training data

# If you want to use own valitadion data do:
clf <- model_structured_data_classifier(max_trials = 10) %>%
  fit(
    train_file,
    "Species",
    validation_data = list(test_file_to_eval, "Species")
  )

# Predict with the best model
(predicted_y <- clf %>% predict(test_file_to_predict))

# Evaluate the best model with testing data
clf %>% evaluate(test_file_to_eval, "Species")

# Get the best trained Keras model, to work with the keras R library
export_model(clf)
```

```
## End(Not run)
```

`model_structured_data_regressor` *AutoKeras Structured Data Regressor Model*

Description

AutoKeras structured data regression class.
To 'fit', 'evaluate' or 'predict', format inputs as:

- x : character or array. If the data is from a csv file, it should be a character specifying the path of the csv file of the training data.
- y : character or array. If the data is from a csv file, it should be a character, which is the name of the target column. Otherwise, it can be single-column or multi-column. The values should all be numerical.

Usage

```
model_structured_data_regressor(
  column_names = NULL,
  column_types = NULL,
  output_dim = NULL,
  loss = "mean_squared_error",
  metrics = NULL,
  name = "structured_data_regressor",
  max_trials = 100,
  directory = tempdir(),
  objective = "val_loss",
  overwrite = TRUE,
  seed = runif(1, 0, 1e+07)
)
```

Arguments

`column_names`	: A list of characters specifying the names of the columns. The length of the list should be equal to the number of columns of the data excluding the target column. Defaults to 'NULL'. If 'NULL', it will obtained from the header of the csv file or the 'data.frame'.
`column_types`	: A list of characters. The names are the column names. The values should either be 'numerical' or 'categorical', indicating the type of that column. Defaults to 'NULL'. If not 'NULL', the 'column_names' need to be specified. If 'NULL', it will be inferred from the data.
`output_dim`	: numeric. The number of output dimensions. Defaults to 'NULL'. If 'NULL', it will infer from the data.

loss	: A Keras loss function. Defaults to use "mean_squared_error".
metrics	: A list of Keras metrics. Defaults to use "mean_squared_error".
name	: character. The name of the AutoModel. Defaults to "structured_data_regressor".
max_trials	: numeric. The maximum number of different Keras Models to try. The search may finish before reaching the ‘max_trials‘. Defaults to ‘100‘.
directory	: character. The path to a directory for storing the search outputs. Defaults to ‘tempdir()‘, which would create a folder with the name of the AutoModel in the current directory.
objective	: character. Name of model metric to minimize or maximize, e.g. "val_accuracy". Defaults to "val_loss".
overwrite	: logical. Defaults to ‘TRUE‘. If ‘FALSE‘, reloads an existing project of the same name if one is found. Otherwise, overwrites the project.
seed	: numeric. Random seed. Defaults to ‘runif(1, 0, 10e6)‘.

Details

Important: The object returned by this function behaves like an R6 object, i.e., within function calls with this object as parameter, it is most likely that the object will be modified. Therefore it is not necessary to assign the result of the functions to the same object.

Value

A non-trained structured data regressor AutokerasModel.

Examples

```
## Not run:
library("keras")

# use the iris dataset as an example
set.seed(8818)
# balanced sample 80% for training
train_idxs <- unlist(by(seq_len(nrow(iris)), iris$Species, function(x) {
  sample(x, length(x) * .8)
}))
train_data <- iris[train_idxs, ]
test_data <- iris[-train_idxs, ]

colnames(iris)
# Sepal.Length will be the interest column to predict

train_file <- paste0(tempdir(), "/iris_train.csv")
write.csv(train_data, train_file, row.names = FALSE)

# file to predict, cant have the response "Species" column
test_file_to_predict <- paste0(tempdir(), "/iris_test_2_pred.csv")
write.csv(test_data[, -1], test_file_to_predict, row.names = FALSE)

test_file_to_eval <- paste0(tempdir(), "/iris_test_2_eval.csv")
```

```
write.csv(test_data, test_file_to_eval, row.names = FALSE)

library("autokeras")

# Initialize the structured data regressor
reg <- model_structured_data_regressor(max_trials = 10) %>% # It tries 10 different models
  fit(train_file, "Sepal.Length") # Feed the structured data regressor with training data

# If you want to use own valitadion data do:
reg <- model_structured_data_regressor(max_trials = 10) %>%
  fit(
    train_file,
    "Sepal.Length",
    validation_data = list(test_file_to_eval, "Sepal.Length")
  )

# Predict with the best model
(predicted_y <- reg %>% predict(test_file_to_predict))

# Evaluate the best model with testing data
reg %>% evaluate(test_file_to_eval, "Sepal.Length")

# Get the best trained Keras model, to work with the keras R library
export_model(reg)

## End(Not run)
```

`model_text_classifier` *AutoKeras Text Classifier Model*

Description

AutoKeras text classification class.
To 'fit', 'evaluate' or 'predict', format inputs as:

- x : array. The input data should be array. The data should be one dimensional. Each element in the data should be a string which is a full sentence.
- y : array. It can be raw labels, one-hot encoded if more than two classes, or binary encoded for binary classification.

Usage

```
model_text_classifier(
  num_classes = NULL,
  multi_label = FALSE,
  loss = NULL,
  metrics = NULL,
  name = "text_classifier",
```

```
  max_trials = 100,
  directory = tempdir(),
  objective = "val_loss",
  overwrite = TRUE,
  seed = runif(1, 0, 1e+07)
)
```

Arguments

num_classes	: numeric. Defaults to 'NULL'. If 'NULL', it will infer from the data.
multi_label	: logical. Defaults to 'FALSE'.
loss	: A Keras loss function. Defaults to use "binary_crossentropy" or "categorical_crossentropy" based on the number of classes.
metrics	: A list of Keras metrics. Defaults to use "accuracy".
name	: character. The name of the AutoModel. Defaults to "text_classifier".
max_trials	: numeric. The maximum number of different Keras Models to try. The search may finish before reaching the 'max_trials'. Defaults to '100'.
directory	: character. The path to a directory for storing the search outputs. Defaults to 'tempdir()', which would create a folder with the name of the AutoModel in the current directory.
objective	: character. Name of model metric to minimize or maximize, e.g. "val_accuracy". Defaults to "val_loss".
overwrite	: logical. Defaults to 'TRUE'. If 'FALSE', reloads an existing project of the same name if one is found. Otherwise, overwrites the project.
seed	: numeric. Random seed. Defaults to 'runif(1, 0, 10e6)'.

Details

Important: The object returned by this function behaves like an R6 object, i.e., within function calls with this object as parameter, it is most likely that the object will be modified. Therefore it is not necessary to assign the result of the functions to the same object.

Value

A non-trained text classifier AutokerasModel.

Examples

```
## Not run:
library("keras")

# Get IMDb dataset
imdb <- dataset_imdb(num_words = 1000)
c(x_train, y_train) %<-% imdb$train
c(x_test, y_test) %<-% imdb$test

# AutoKeras procceses each text data point as a character vector,
```

```
# i.e., x_train[[1]] "<START> this film was just brilliant casting..",
# so we need to transform the dataset.
word_index <- dataset_imdb_word_index()
word_index <- c(
  "<PAD>", "<START>", "<UNK>", "<UNUSED>",
  names(word_index)[order(unlist(word_index))]
)
x_train <- lapply(x_train, function(x) {
  paste(word_index[x + 1], collapse = " ")
})
x_test <- lapply(x_test, function(x) {
  paste(word_index[x + 1], collapse = " ")
})

x_train <- array(unlist(x_train))
x_test <- array(unlist(x_test))
y_train <- matrix(y_train, ncol = 1)
y_test <- matrix(y_test, ncol = 1)

library("autokeras")

# Initialize the text classifier
clf <- model_text_classifier(max_trials = 10) %>% # It tries 10 different models
  fit(x_train, y_train) # Feed the text classifier with training data

# If you want to use own valitadion data do:
clf <- model_text_classifier(max_trials = 10) %>%
  fit(
    x_train,
    y_train,
    validation_data = list(x_test, y_test)
  )

# Predict with the best model
(predicted_y <- clf %>% predict(x_test))

# Evaluate the best model with testing data
clf %>% evaluate(x_test, y_test)

# Get the best trained Keras model, to work with the keras R library
export_model(clf)

## End(Not run)
```

---

model_text_regressor *AutoKeras Text Regressor Model*

Description

AutoKeras text regression class.
To ‘fit‘, ‘evaluate‘ or ‘predict‘, format inputs as:

- x : array. The input data should be array. The data should be one dimensional. Each element in the data should be a string which is a full sentence.
- y : array. The targets passing to the head would have to be array or data.frame. It can be single-column or multi-column. The values should all be numerical.

Usage

```
model_text_regressor(
  output_dim = NULL,
  loss = "mean_squared_error",
  metrics = NULL,
  name = "text_regressor",
  max_trials = 100,
  directory = tempdir(),
  objective = "val_loss",
  overwrite = TRUE,
  seed = runif(1, 0, 1e+07)
)
```

Arguments

output_dim	: numeric. The number of output dimensions. Defaults to ‘NULL‘. If ‘NULL‘, it will infer from the data.
loss	: A Keras loss function. Defaults to use "mean_squared_error".
metrics	: A list of Keras metrics. Defaults to use "mean_squared_error".
name	: character. The name of the AutoModel. Defaults to "text_regressor".
max_trials	: numeric. The maximum number of different Keras Models to try. The search may finish before reaching the ‘max_trials‘. Defaults to ‘100‘.
directory	: character. The path to a directory for storing the search outputs. Defaults to ‘tempdir()‘, which would create a folder with the name of the AutoModel in the current directory.
objective	: character. Name of model metric to minimize or maximize, e.g. "val_accuracy". Defaults to "val_loss".
overwrite	: logical. Defaults to ‘TRUE‘. If ‘FALSE‘, reloads an existing project of the same name if one is found. Otherwise, overwrites the project.
seed	: numeric. Random seed. Defaults to ‘runif(1, 0, 10e6)‘.

Details

Important: The object returned by this function behaves like an R6 object, i.e., within function calls with this object as parameter, it is most likely that the object will be modified. Therefore it is not necessary to assign the result of the functions to the same object.

Value

A non-trained text regressor AutokerasModel.

Examples

```
## Not run:
library("keras")

# Get IMDb dataset
imdb <- dataset_imdb(num_words = 1000)
c(x_train, y_train) %<-% imdb$train
c(x_test, y_test) %<-% imdb$test

# AutoKeras procceses each text data point as a character vector,
# i.e., x_train[[1]] "<START> this film was just brilliant casting..",
# so we need to transform the dataset.
word_index <- dataset_imdb_word_index()
word_index <- c(
  "<PAD>", "<START>", "<UNK>", "<UNUSED>",
  names(word_index)[order(unlist(word_index))]
)
x_train <- lapply(x_train, function(x) {
  paste(word_index[x + 1], collapse = " ")
})
x_test <- lapply(x_test, function(x) {
  paste(word_index[x + 1], collapse = " ")
})

x_train <- array(unlist(x_train))
x_test <- array(unlist(x_test))
y_train <- matrix(y_train, ncol = 1)
y_test <- matrix(y_test, ncol = 1)

library("autokeras")

# Initialize the text regressor
reg <- model_text_regressor(max_trials = 10) %>% # It tries 10 different models
  fit(x_train, y_train) # Feed the text regressor with training data

# If you want to use own valitadion data do:
reg <- model_text_regressor(max_trials = 10) %>%
  fit(
    x_train,
    y_train,
    validation_data = list(x_test, y_test)
  )

# Predict with the best model
(predicted_y <- reg %>% predict(x_test))

# Evaluate the best model with testing data
reg %>% evaluate(x_test, y_test)
```

```
# Get the best trained Keras model, to work with the keras R library
export_model(reg)

## End(Not run)
```

---

predict *Model Predictions*

Description

Predict the output for a given testing data.

Usage

```
## S3 method for class 'AutokerasModel'
predict(object, x, batch_size = 32, ...)
```

Arguments

object	: A trained AutokerasModel instance.
x	: Any allowed types according to the input node. Testing data. Check corresponding AutokerasModel help to note how it should be provided.
batch_size	: numeric. Defaults to ‘32‘.
...	: Unused.

Value

A one-column matrix with the predicted values as rows.

Examples

```
## Not run:
library("keras")

# use the MNIST dataset as an example
mnist <- dataset_mnist()
c(x_train, y_train) %<-% mnist$train
c(x_test, y_test) %<-% mnist$test

library("autokeras")

# Initialize the image classifier
clf <- model_image_classifier(max_trials = 10) %>% # It tries 10 different models
  fit(x_train, y_train) # Feed the image classifier with training data

# Predict with the best model
```

```
(predicted_y <- clf %>% predict(x_test))

# Evaluate the best model with testing data
clf %>% evaluate(x_test, y_test)

# Get the best trained Keras model, to work with the keras R library
export_model(clf)

## End(Not run)
```

`save_model` *Save Model*

Description

Save the AutoKeras Model.
Make sure that ‘directory‘ and ‘name‘ values are set when creating the model, and that the ‘directory‘ is not a temporary folder.

Usage

```
save_model(autokeras_model, filename)
```

Arguments

`autokeras_model`	: A trained AutokerasModel instance.
`filename`	: A character string naming a file to save the model.

Value

None

Examples

```
## Not run:
library("keras")

# use the MNIST dataset as an example
mnist <- dataset_mnist()
c(x_train, y_train) %<-% mnist$train
c(x_test, y_test) %<-% mnist$test

library("autokeras")

# Initialize the image classifier
clf <- model_image_classifier(
  directory = ".",
```

```
  name = "autokeras_mnist",
  max_trials = 10
) %>% # It tries 10 different models
  fit(x_train, y_train) # Feed the image classifier with training data

# Predict with the best model
(predicted_y <- clf %>% predict(x_test))

# Evaluate the best model with testing data
clf %>% evaluate(x_test, y_test)

# Get the best trained Keras model, to work with the keras R library
export_model(clf)

# Save the AutoKeras model.
# Make sure that `directory` and `name` were set when creating the model.
save_model(clf, "my_model.pkl")

# And load it again
new_clf <- load_model("my_model.pkl")

## End(Not run)
```

Index