
project-template Documentation

Release 0.0.1

Vighnesh Birodkar

Feb 17, 2020

1	Quick Start with the minimization toolkit	3
1.1	Download the minimization toolkit code:	3
2	User guide: start minimizing your ML model	5
2.1	The GeneralizeToRepresentative class	5
2.2	How to use GeneralizeToRepresentative	5
3	ai-minimization-toolkit API	7
3.1	GeneralizeToRepresentative	7
4	Getting started	11
5	User Guide	13
6	API Documentation	15
7	Examples	17
	Index	19

This project provides data minimization capabilities for any scikit-learn model.

Quick Start with the minimization toolkit

1.1 Download the minimization toolkit code:

Clone the ai-minimization-toolkit repository:

```
$ git clone https://github.com/IBM/ai-minimization-toolkit.git
```

Or download using pip:

```
pip install ai-minimization-toolkit==0.0.1
```

User guide: start minimizing your ML model

2.1 The GeneralizeToRepresentative class

The main class, `minimization.GeneralizeToRepresentative`, is a scikit-learn compatible Transformer, that receives an existing estimator and labeled training data, and learns the generalizations that can be applied to any newly collected data for analysis by the original model.

- at `fit`, the generalizations are learned from `X` and `y`;
- at `transform`, `X` will be transformed, using the generalizations learned during `fit`;
- `fit_transform` will both learn the generalizations and then apply them to `X`.

It is also possible to export the generalizations as feature ranges, for example to create forms for data collection.

The current implementation supports only numeric features, so any categorical features must be transformed to a numeric representation before using this class.

2.2 How to use GeneralizeToRepresentative

Start by training your machine learning model. In this example, we will use a `sklearn.tree.DecisionTreeClassifier`, but any scikit-learn model can be used. We will use the iris dataset in our example.

```
from sklearn import datasets
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier

dataset = datasets.load_iris()
X_train, X_test, y_train, y_test = train_test_split(dataset.data, dataset.target,
    ↳test_size=0.2)

base_est = DecisionTreeClassifier()
base_est.fit(X_train, y_train)
```

Now create the `minimization.GeneralizeToRepresentative` transformer and train it. Supply it with the original model and the desired target accuracy. The training process may receive the original labeled training data or the model's predictions on the data.

```
predictions = base_est.predict(X_train)
gen = GeneralizeToRepresentative(base_est, target_accuracy=0.9)
gen.fit(X_train, predictions)
```

Now use the transformer to transform new data, for example the test data.

```
transformed = gen.transform(X_test)
```

The transformed data has the same columns and formats as the original data, so it can be used directly to derive predictions from the original model.

```
new_predictions = base_est.predict(transformed)
```

To export the resulting generalizations, retrieve the Transformer's `_generalize` parameter.

```
generalizations = base_est._generalize
```

The returned object has the following structure:

```
{
  ranges:
  {
    list of (<feature name>: [<list of values>])
  },
  untouched: [<list of feature names>]
}
```

For example:

```
{
  ranges:
  {
    age: [21.5, 39.0, 51.0, 70.5],
    education-years: [8.0, 12.0, 14.5]
  },
  untouched: ["occupation", "marital-status"]
}
```

Where each value inside the range list represents a cutoff point. For example, for the `age` feature, the ranges in this example are: `<21.5`, `21.5–39.0`, `39.0–51.0`, `51.0–70.5`, `>70.5`. The `untouched` list represents features that were not generalized, i.e., their values should remain unchanged.

3.1 GeneralizeToRepresentative

<code>GeneralizeToRepresentative</code> (<code>estimator</code> , ...)	A transformer that generalizes data to representative points.
--	---

3.1.1 minimization.GeneralizeToRepresentative

class `minimization.GeneralizeToRepresentative` (*estimator=None*, *target_accuracy=0.998*,
features=None, *cells=None*)

A transformer that generalizes data to representative points.

Learns data generalizations based on an original model's predictions and a target accuracy. Once the generalizations are learned, can receive one or more data records and transform them to representative points based on the learned generalization.

An alternative way to use the transformer is to supply `cells` and `features` in `init` or `set_params` and those will be used to transform data to representatives. In this case, `fit` must still be called but there is no need to supply it with `X` and `y`, and there is no need to supply an existing `estimator` to `init`.

In summary, either `estimator` and `target_accuracy` should be supplied or `cells` and `features` should be supplied.

Parameters

estimator [estimator, optional] The original model for which generalization is being performed. Should be pre-fitted.

target_accuracy [float, optional] The required accuracy when applying the base model to the generalized data. Accuracy is measured relative to the original accuracy of the model.

features [list of str, optional] The feature names, in the order that they appear in the data.

cells [list of object, optional] The cells used to generalize records. Each cell must define a range or subset of categories for each feature, as well as a representative value for each feature.

This parameter should be used when instantiating a transformer object without first fitting it.

Attributes

cells_ [list of object] The cells used to generalize records, as learned when calling fit.

nep_ [float] The NCP (information loss) score of the resulting generalization, as measured on the training data.

generalizations_ [object] The generalizations that were learned (actual feature ranges).

__init__ (*self*, *estimator=None*, *target_accuracy=0.998*, *features=None*, *cells=None*)
Initialize self. See help(type(self)) for accurate signature.

fit (*self*, *X=None*, *y=None*)
Learns the generalizations based on training data.

Parameters

X [{array-like, sparse matrix}, shape (n_samples, n_features), optional] The training input samples.

y [array-like, shape (n_samples,), optional] The target values. An array of int. This should contain the predictions of the original model on X.

Returns

X_transformed [ndarray, shape (n_samples, n_features)] The array containing the representative values to which each record in X is mapped.

fit_transform (*self*, *X=None*, *y=None*)
Learns the generalizations based on training data, and applies them to the data.

Parameters

X [{array-like, sparse matrix}, shape (n_samples, n_features), optional] The training input samples.

y [array-like, shape (n_samples,), optional] The target values. An array of int. This should contain the predictions of the original model on X.

Returns

self [object] Returns self.

get_params (*self*, *deep=True*)
Get parameters for this estimator.

Parameters

deep [boolean, optional] If True, will return the parameters for this estimator and contained subobjects that are estimators.

Returns

params [mapping of string to any] Parameter names mapped to their values.

set_params (*self*, ***params*)
Set the parameters of this estimator.

Returns

self [object] Returns self.

transform (*self*, *X*)
Transforms data records to representative points.

Parameters

X [{array-like, sparse-matrix}, shape (n_samples, n_features)] The input samples.

Returns

X_transformed [ndarray, shape (n_samples, n_features)] The array containing the representative values to which each record in X is mapped.

CHAPTER 4

Getting started

Information regarding how to use this project.

CHAPTER 5

User Guide

Narrative documentation.

CHAPTER 6

API Documentation

API documentation.

CHAPTER 7

Examples

A set of examples. It complements the [User Guide](#).

Symbols

`__init__()` (*minimization.GeneralizeToRepresentative* method), 8

F

`fit()` (*minimization.GeneralizeToRepresentative* method), 8

`fit_transform()` (*minimization.GeneralizeToRepresentative* method), 8

G

`GeneralizeToRepresentative` (class in *minimization*), 7

`get_params()` (*minimization.GeneralizeToRepresentative* method), 8

S

`set_params()` (*minimization.GeneralizeToRepresentative* method), 8

T

`transform()` (*minimization.GeneralizeToRepresentative* method), 8