UNIVERSIDADE CATÓLICA PORTUGUESA

BRAGA

Machine Learning

Session 4 - PL

Preprocessing

Ciência de Dados Aplicada

2023/2024

Machine Learning From Scratch

- GitHub Repository Development:
  - We'll be building a machine learning package from scratch, primarily using NumPy.

- Today's Implementations:
  - **Mean Imputer:** Implementing a mean imputer to handle missing values.
  - **Standardization Method:** Implementing a data scaling method to standardize our dataset.
  - **Variance Threshold:** Implementing a variance threshold method for feature selection.

Package Overview

- **setup.py** file: configuration file for packaging and distribution.
- Package organization:
  - **'src'** directory: contains the source code of the Python package;
  - **'ml_from_scratch'** directory: contains the Python package code. It includes the modules, sub-packages, and any other necessary files that define the functionality of the package.
  - **'__init__.py'** files: special Python files that indicate to Python that a directory should be considered a package or a module.
  - Directories inside 'ml_from_scratch': represent sub-packages or modules within the package.

Transformers

- In our package methods that transfor the data will follow the structure of a Transformer.

- **Transformer Architecture:**
  - parameters - a set of user defined parameters;
  - attributes / estimated parameters - parameters/attributes estimated from the data;
  - fit - a method responsible for estimating parameters from the data;
  - transform – a method responsible for transforming the data.

Class Transformer

- **Class Transformer(ABC):**
  - Abstract class for all transformers;
  - Attributes:
    - _fitted – whether the transformer is fitted or not.
  - Methods:
    - fit – method that receives x and y and fits the transformer; sets _fitted to True if the transformer is successfully fitted.
    - _fit – abstract method that implements the fit logic of each transformer.
    - transform – method that receives x and y and transforms the data based on the estimated parameters; checks if the transformer is fitted before transforming the data.
    - _transform – abstract method that implements the transform logic of each transformer.
    - fit_transform – applies fit and then transform.
    - fitted – returns whether the transformer is fitted or not.

Class MeanImputer

- **Class MeanImputer(Transformer):**
  - Imputes missing values with the column mean.
  - Attributes:
    - means – the mean of each column in the dataset.
  - Methods:
    - _fit – computes "means" from the data.
    - _transform – imputes missing values using the "means" attribute.

Class Standardization

- **Class Standardization(Transformer):**
  - Centers the data around mean 0 and standard deviation 1.
  - Attributes:
    - mean – the mean of each column in the dataset.
    - std – the standard deviation of each column in the dataset.
  - Methods:
    - _fit – computes "mean" and "std" from the data.
    - _transform - standardizes the data using the "mean" and "std" attributes.
      - X= (X-mean) / std
    - inverse_transform - transforms the standardized data back to its original scale.
      - X = X * std + mean

Class VarianceThreshold

- **Class VarianceThreshold(Transformer):**
  - Removes features with variance bellow a threshold.
  - Parameters:
    - threshold – cut-off value.
  - Attributes:
    - variances – the variance of each feature.
    - selected_features – features to keep, i.e., features with variance greater than "threshold".
  - Methods:
    - _fit – computes "variances" and "selected_features" from the data.
    - _transform – selects the features accordinf to "selected_features".

Exercise 1:

- Create a new module called "mode_imputer.py" inside the "imputat ion" sub-package and implement the ModeImputer class.

- **Class ModeImputer(Transformer):**
  - Imputes missing values with the column mode.
  - Attributes:
    - modes – the mode of each column in the dataset.
  - Methods:
    - _fit – computes "modes" from the data.
    - _transform – imputes missing values using the "modes" attribute.

- Hint: sometimes the mode is not a single value!

Exercise 2:

- Create a new module called "normalization.py" inside the "scaling" sub-package and implement the Normalization class.

- **Class Normalization(Transformer):**
  - Scales data between 0 and 1;
  - Attributes:
    - min - the minimum value of each column in the dataset.
    - max - the maximum value of each column in the dataset.
  - Methods:
    - _fit – Computes the minimum and maximum values from the data.
    - _transform – scales the data between 0 and 1 using the computed minimum and maximum values.
    - inverse_transform - inverse operation to rescale the data back to its original range.

$$x_{\text{norm}} = \frac{x - \min(x)}{\max(x) - \min(x)}$$

Exercise 3:

- Create a new module called "select_percentile.py" inside the "feature_selection" sub-package and implement the SelectPercentile class.

- **Class SelectPercentile(Transformer):**
  - Selects features based on percentile of the highest scores.
  - Arguments:
    - k – number of features to select.
    - score_func - function used to compute scores for each feature. By default "f_classif" available in the "statistics" sub-package.
  - Attributes:
    - scores - scores computed for each feature.
    - selected_features - indices of selected features.
  - Methods:
    - _fit – compute scores from the data and defines the "k" "selected_features".
    - _transform – select the features using "selected_features".