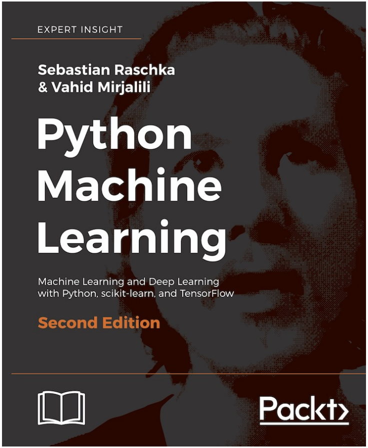
Machine Learning I (DATS 6202)
Data Preprocessing

Yuxiao Huang

Data Science, Columbian College of Arts & Sciences
George Washington University
yuxiaohuang@gwu.edu

August 27, 2018

Reference



Picture courtesy of the website of the book code repository and info resource

Reference

- This set of slices is an excerpt of the book by Raschka and Mirjalili, with some trivial changes by the creator of the slides
- Please find the reference to and website of the book below:
  - *Raschka S. and Mirjalili V. (2017). Python Machine Learning. 2nd Edition.*
  - `https://sebastianraschka.com/books.html`
- Please find the website of the book code repository and info resource below:
  - `https://github.com/rasbt/python-machine-learning-book-2nd-edition`

Overview

1. Dealing with missing data
2. Handling categorical data
3. Partitioning training and testing data
4. Bringing feature onto the same scale

Dealing with missing data

- Missing data abound in reality, for many reasons
  - human error
  - equipment error
  - left blank intentionally (e.g., in a survey)
- Missing data could be shown as
  - blank space
  - placeholder NA or N/A (Not Applicable)
  - placeholder NaN or nan (Not A Number)
  - ? (as in UCI data)

Dealing with missing data

- Eliminating samples or features with missing values
    - Often infeasible. Why?
- Imputing missing values
    - mean
    - median
    - mode (useful for categorical features). Why?

Handling categorical data

- Most machine learning methods work (better) on numerical data
- However, real-world data usually have categorical features
  - size: XL, L, M, S
  - color: red, blue, green

Handling categorical data

- Ordinal feature
    - categorical feature that can be sorted or ordered
    - size: XL $>$ L $>$ M $>$ S
- Nominal feature
    - categorical feature that cannot be sorted or ordered
    - color: red, blue, green
- We need to distinguish between nominal and ordinal features. Why?

Mapping ordinal features

- Converting the categorical string values into integers
- The features must be ordinal
- Must preserve the order
  - XL : 3
  - L : 2
  - M : 1
  - S : 0

Encoding class labels

- Converting the categorical string values into integers
- The class can be ordinal or nominal
- No need to preserve the order
  - class 1 : 3
  - class 2 : 2
  - class 3 : 1

Mapping nominal features

- Converting the categorical string values into integers
  - red : 0
  - blue : 1
  - green : 2

Mapping nominal features

- Converting the categorical string values into integers
  - red : 0
  - blue : 1
  - green : 2
- This is one of the most common mistakes in dealing with nominal features!
  - Since we assume red $<$ blue $<$ green

One-hot encoding

- Transform the feature into feature-value pairs
  - color = red
  - color = blue
  - color = green
- Each feature-value pair is binary (e.g., 0 or 1)
- For the original feature, only one feature-value is 1 at a time

Partitioning training and testing data

- On the one hand
    - the size of testing set $\uparrow$
    - information withheld from the learning algorithm $\downarrow$
- On the other hand
    - the size of testing set $\downarrow$
    - estimation of the generalization error $\uparrow$

Partitioning training and testing data

- We need to balance this trade-off
- The most commonly used splits are
  - 60:40
  - 70:30
  - 80:20

Bringing feature onto the same scale

- Majority of machine learning and optimization algorithms behave much better if features are on the same scale
- There are two common approaches to bringing different features onto the same scale
    - Normalization
    - Standardization

(min-max) Normalization

- To normalize our data, we can simply apply the min-max scaling to each feature column, where the new value $x^{(i)}_{norm}$ of a sample $x^{(i)}$:

$$x^{(i)}_{norm} = \frac{x^{(i)} - \mathbf{x}_{min}}{\mathbf{x}_{max} - \mathbf{x}_{min}}$$

- Here:
  - $x^{(i)}$ is one of the original values of feature x
  - $\mathbf{x}_{min}$ / $\mathbf{x}_{max}$ is the min / max of all the original values
  - $x^{(i)}_{norm}$ is the value after normalization

Standardization

- Standardization maintains useful information about outliers and makes the algorithm less sensitive to them in contrast to min-max scaling, which scales the data to a limited range of values:

$$x_{std}^{(i)} = \frac{x^{(i)} - \mu_x}{\sigma_x}$$

- Here:
  - $x^{(i)}$ is one of the original values of feature x
  - μ_x is the mean of feature x
  - σ_x is the standard deviation of feature x