Lecture 6: Preprocessing

Credit

The following slides are based on:

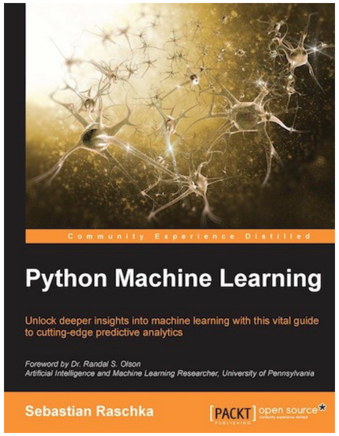
Scikit-learn, Preprocessing data

http://scikit-learn.org/stable/modules/preprocessing.html

FEATURE SCALING WITH SCIKIT-LEARN

http://benalexkeen.com/feature-scaling-with-scikit-learn/

And this book


Community Experience Distilled
Python Machine Learning
Unlock deeper insights into machine learning with this vital guide to cutting-edge predictive analytics
Foreword by Dr. Randal S. Olson
Artificial Intelligence and Machine Learning Researcher, University of Pennsylvania
Sebastian Raschka
PACKT open source

Agenda

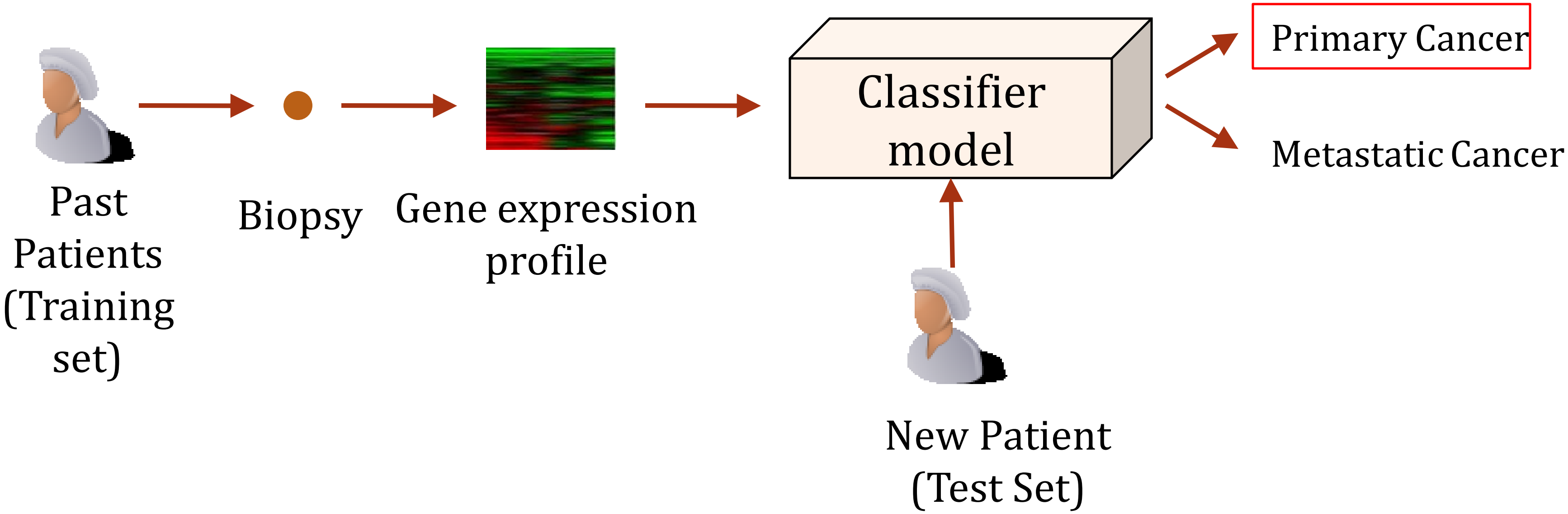
- Preprocessing structured data
- We will use Pandas
- We will also use Scikit-learn toolkit which is a Python library for conventional machine learning tasks.

Topics

- Missing Data
- Converting features
- Converting class labels
- Standardization, Scaling, Normalization

Machine Learning

- Big picture



Past Patients (Training set)

Biopsy

Gene expression profile

Classifier model

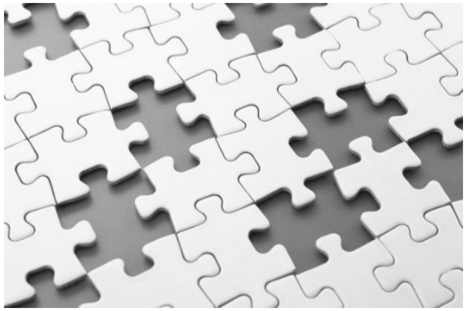
Primary Cancer

Metastatic Cancer

New Patient (Test Set)

Missing Data

- Missing data is quite common in the medical domain
- It needs to be handled carefully



Preprocessing – Missing Values

- Remove missing data
  - We might lose too much valuable data
- Imputing data
  - Mean
  - Median
  - …
- Replacing with normal clinical values
- Best approach is to inspect your data
  - What percentage of data is missing?
- See Preprocessing.ipynb Notebook on Canvas

Features

- There are several different types of structured features
  - Categorical
    - Nominal
    - Ordinal
  - Continuous

- We will look at unstructured data later
  - Images, text, time series, ….

Data and Scikit-learn

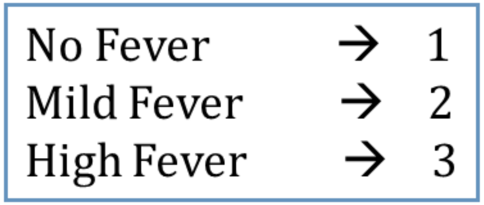
- Scikit input: Everything has to be numbers only (float or integer)
- You can convert a DataFrame to a NumPy array using .values attribute of the DataFrame object
  - But this might not exactly work with categorical values and class labels
  - You have to convert them manually
  - You could also use additional packages to help in transforming data, see here

Handling Categorical Data

- Categorical data
  - Ordinal: can be sorted or ordered (e.g. T-shirt size: XL > L > M) or (Fever: none < mild < high)
  - Nominal: doesn’t imply any order (e.g. T-short color) or (Gender)

Ordinal Features

- You need to define the mapping manually (see Notebook)



No Fever	→	1
Mild Fever	→	2
High Fever	→	3

Class Labels

- Many machine learning libraries require that class labels are encoded as integer values

Tumor	→	0
Non-Tumor	→	1

Class Labels

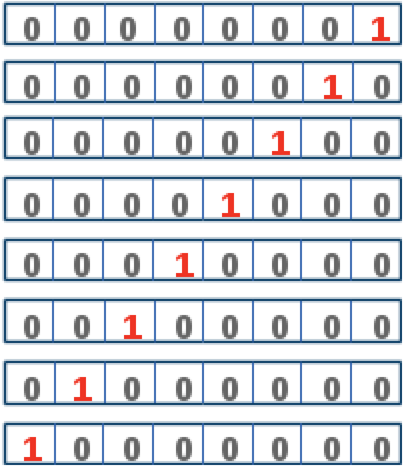
- Most estimators for classification in scikit-learn convert class labels to integers internally
- Still good practice to provide class labels as integer arrays to avoid technical glitches.
  - Use LabelEncoder() from sklearn.preprocessing
  - It uses values between 0 and n_classes -1 to encode the labels.
  - E.g. First class= 0, second class = 1, ….

Nominal Features

- Using LabelEncoder()? Would that work?

Nominal Features

- You should use one-hot encoding
  - get_dummies method implemented in pandas



0	0	0	0	0	0	0	1
0	0	0	0	0	0	1	0
0	0	0	0	0	1	0	0
0	0	0	0	1	0	0	0
0	0	0	1	0	0	0	0
0	0	1	0	0	0	0	0
0	1	0	0	0	0	0	0
1	0	0	0	0	0	0	0

Next Steps

- After converting everything into integers and floats, we are not done yet!
- We still need to transform the values

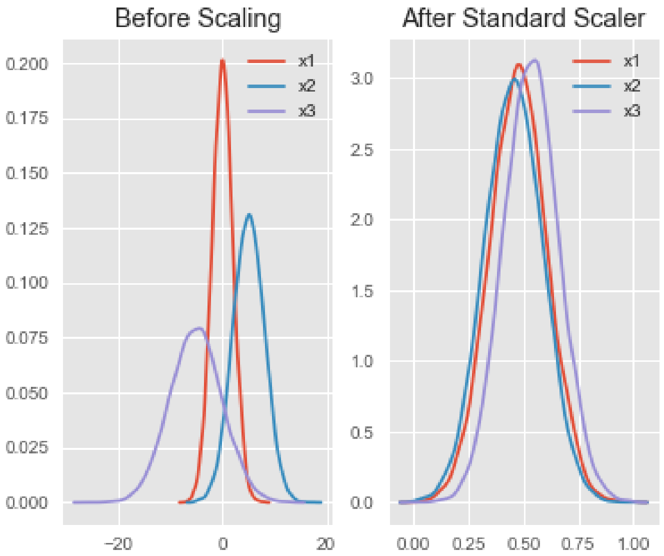
Feature Transformation

- Standardization
- Scaling
- Normalization

Standardization(or Z-score Normalization)

- Removing the mean value of each feature, then scale each feature by it by its standard deviation (all new features will have a zero mean and unit variance)

$$x_{new} = \frac{x - \mu}{\sigma}$$



http://benalexkeen.com/feature-scaling-with-scikit-learn/

Reason for Standardization

- Not only important if we are comparing measurements that have different units, but it is also a general requirement for many machine learning algorithms.

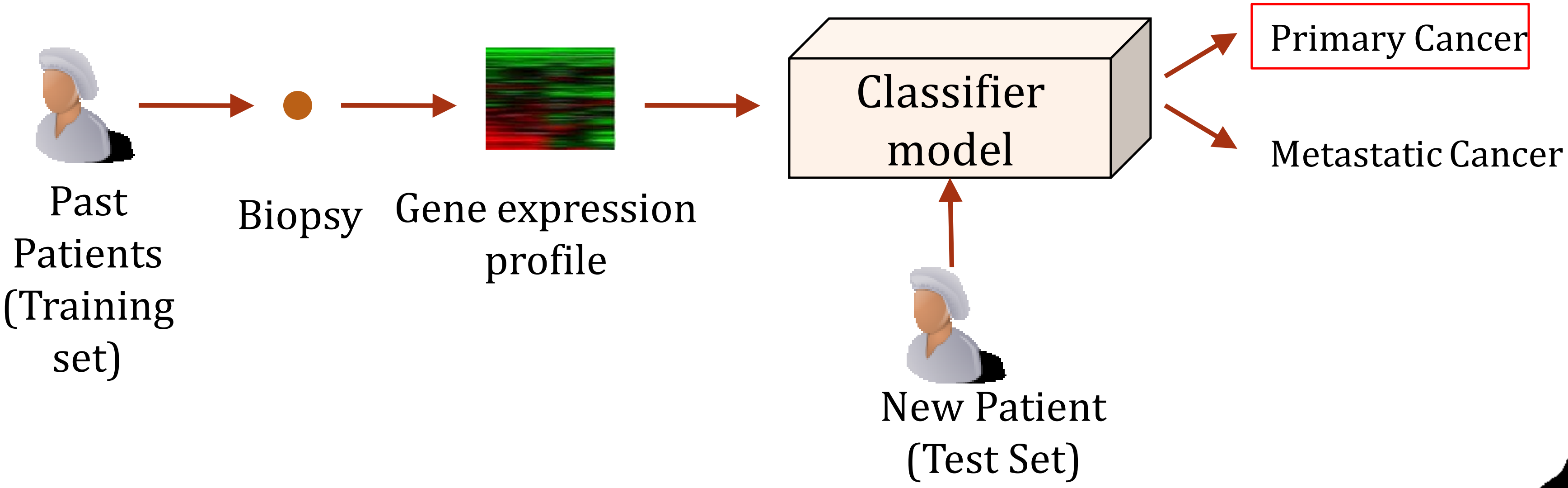
Reason for Standardization

- We will later see that many elements used in the objective function of a learning algorithm (such as the RBF kernel of Support Vector Machines) assume that all features are centered around zero and have variance in the same order.
- If a feature has a variance that is orders of magnitude larger than others, it might dominate the objective function and make the estimator unable to learn from other features correctly as expected (as in gradient descent).

http://scikit-learn.org/stable/modules/preprocessing.html

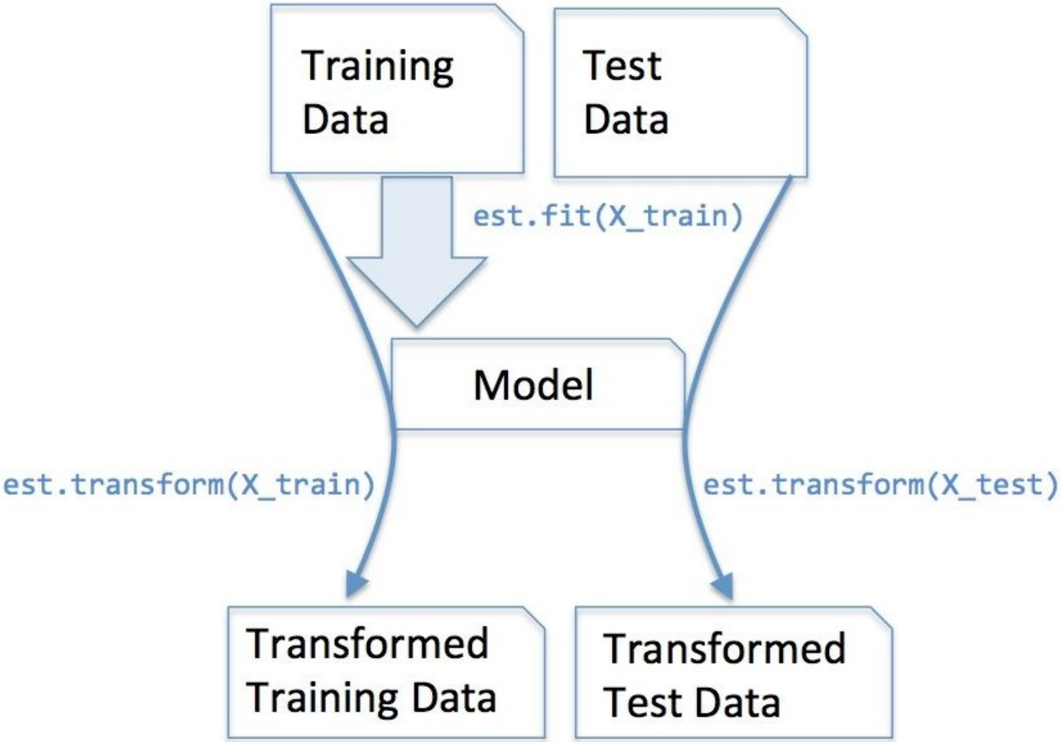
Transforming Features

- Machine learning pipeline typically involves a training stage and a testing stage.



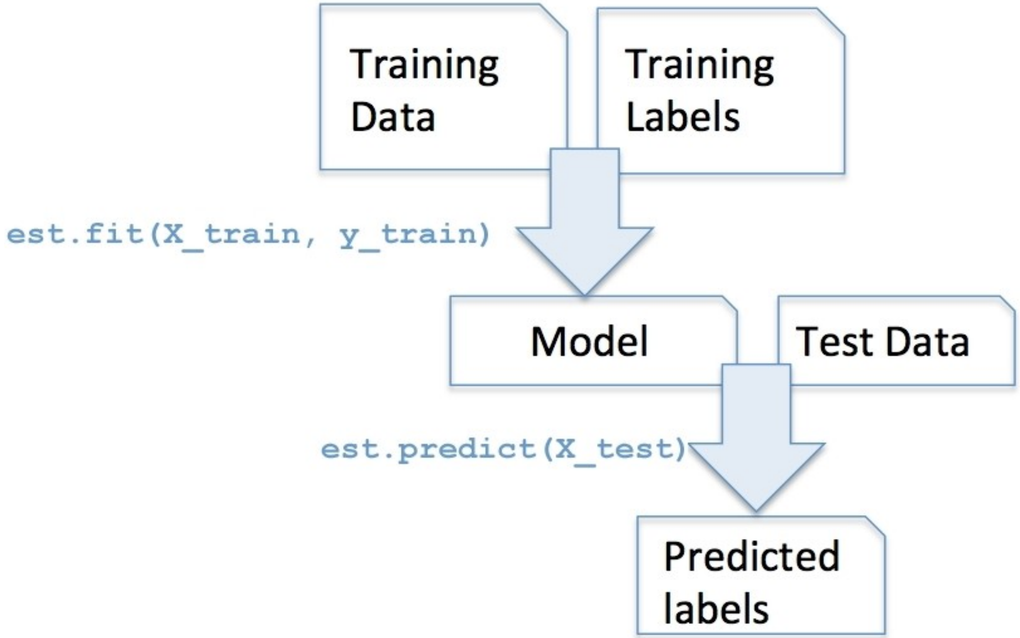
Understanding scikit-learn API

- A two step process
  1. Fit
  2. Transform



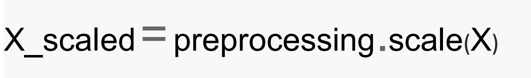
Understanding scikit-learn API

- In case of classifiers
  1. Fit
  2. Predict

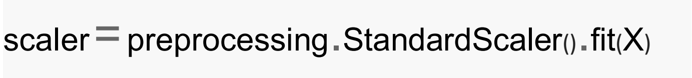


Standardization in Scikit-learn

- Two different methods (old API)



```
X_scaled = preprocessing.scale(X)
```



```
scaler = preprocessing.StandardScaler().fit(X)
```

✓

- New API
  - Most transformation operations now can be done using a single operator in scikit-learn, the ColumnTarnsformer.

http://scikit-learn.org/stable/modules/preprocessing.html

Scaling

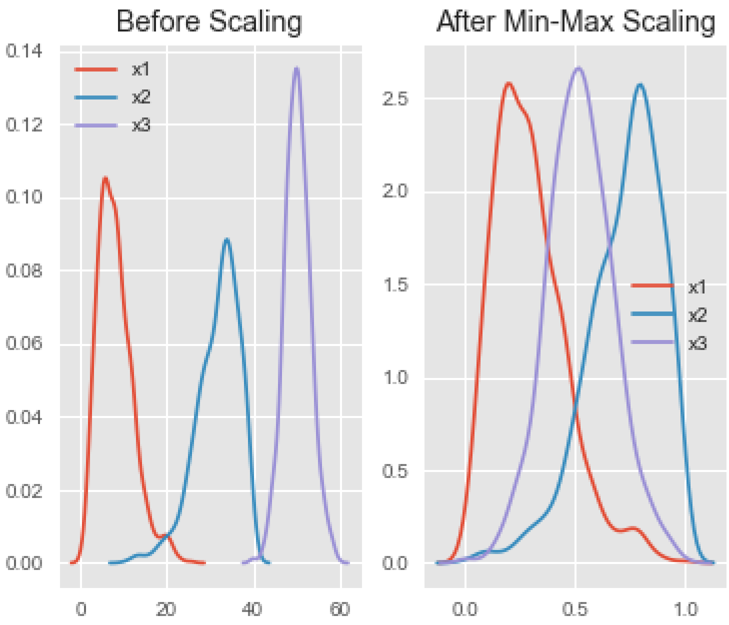
- Alternative to standardization, using min and max scaling.

$$x_{new} = \frac{x - x_{min}}{x_{max} - x_{min}}$$

```
>>> min_max_scaler = preprocessing.MinMaxScaler()

>>> X_train_minmax = min_max_scaler.fit_transform(X_train)
```

```
>>> X_test_minmax = min_max_scaler.transform(X_test)
```



http://benalexkeen.com/feature-scaling-with-scikit-learn/

http://scikit-learn.org/stable/modules/preprocessing.html

Scaling: Sparse Data

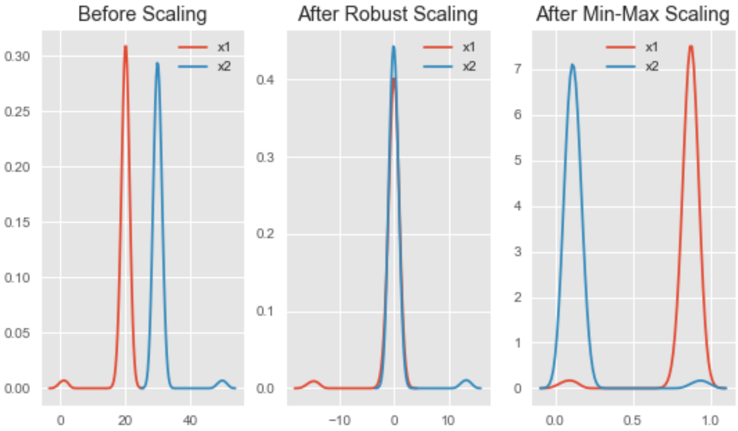
- Centering sparse data would destroy the sparseness structure in the data, and thus rarely is a sensible thing to do.
- However, it can make sense to scale sparse inputs, especially if features are on different scales.

http://scikit-learn.org/stable/modules/preprocessing.html

Scaling: Outlier

- If your data contains many outliers, scaling using the mean and variance is likely to not work very well.
- In these cases, you can use robust_scale and RobustScaler as drop-in replacements instead. They use more robust estimates for the center and range of your data.

$$\frac{x_i - Q_1(x)}{Q_3(x) - Q_1(x)}$$



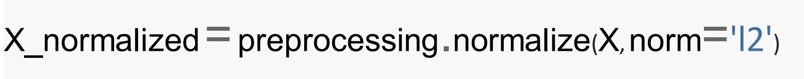
http://benalexkeen.com/feature-scaling-with-scikit-learn/

http://scikit-learn.org/stable/modules/preprocessing.html

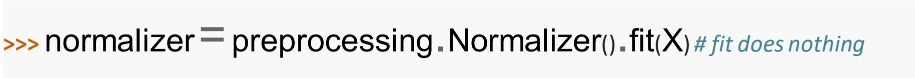
Normalization

- Normalization is the process of scaling individual samples to have unit norm (Each point within 1 unit of the origin).
- often used in text classification.

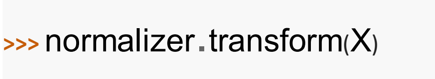
$$\frac{x_i}{\sqrt{x_i^2 + y_i^2 + z_i^2}}$$



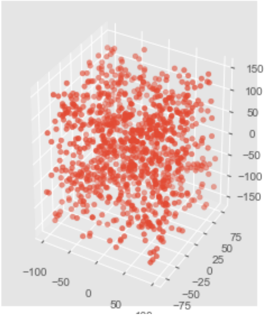
```
X_normalized = preprocessing.normalize(X, norm='l2')
```

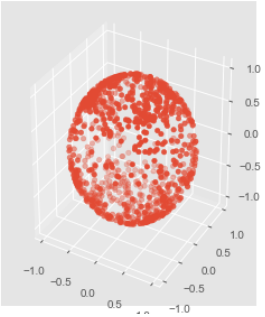


```
>>> normalizer = preprocessing.Normalizer().fit(X) # fit does nothing
```



```
>>> normalizer.transform(X)
```





http://benalexkeen.com/feature-scaling-with-scikit-learn/

http://scikit-learn.org/stable/modules/preprocessing.html

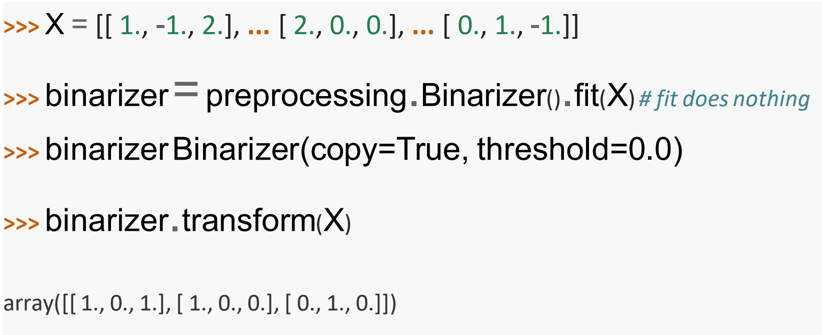
Binarization

- Feature binarization is the process of thresholding numerical features to get boolean values.
- It is common among the text processing community to use binary feature values.

http://scikit-learn.org/stable/modules/preprocessing.html

Binarization Example

- Note that the threshold can be adjusted.



```
>>> X = [[ 1., -1., 2.], ... [ 2., 0., 0.], ... [ 0., 1., -1.]]

>>> binarizer = preprocessing.Binarizer().fit(X) # fit does nothing
>>> binarizer Binarizer(copy=True, threshold=0.0)

>>> binarizer.transform(X)


array([[ 1., 0., 1.], [ 1., 0., 0.], [ 0., 1., 0.]])
```

http://scikit-learn.org/stable/modules/preprocessing.html

ColumnTransformer

- Most of these operations now can be done using a single operator in scikit-learn, the ColumnTarnsformer.

- See the example notebook.

What to do, when?

- It depends.
- Standardizing either input or target variables tends to make the training process better behaved by improving the numerical condition
  - For example, with neural networks, typically it is better to center your data.
  - With some algorithms such as kNN, the result depends on the magnitude of each feature, so again, necessary to do standardizing.
  - It might not matter much for tree-based methods

Algorithms where feature standardizing matters

- k-nearest neighbors with a Euclidean distance measure if we want all features to contribute equally
- k-means (same logic as k-nearest neighbors)
- logistic regression, SVMs, perceptrons, neural networks etc. if you are using gradient descent/ascent-based optimization, otherwise some weights will update much faster than others
- linear discriminant analysis, principal component analysis, kernel principal component analysis

http://sebastianraschka.com/Articles/2014_about_feature_scaling.html

Final Words

- When in doubt, just standardize the data, it shouldn’t hurt.
- However, this doesn’t mean that Min-Max scaling is not useful at all!
- A popular application is image processing, where pixel intensities have to be scaled to fit within a certain range (i.e., 0 to 255 for the RGB color range).

Preprocessing – Pandas Library

- For preprocessing, you can also use the pandas library
  - Data will be stored in a DataFrame object
- Scikit-learn was developed to work with NumPy arrays
  - Still, sometimes it can be convenient to preprocess data using Pandas as a DataFrame
  - You can access it as a NumPy array using .values attribute of the DataFrame object
    - Scikit input: remember, everything has to be numbers only (float or integer)