wbs
WARWICK BUSINESS SCHOOL
THE UNIVERSITY OF WARWICK

For the Change Makers

Advanced Programming for Data Science

Week 9: Model Building and Tuning

Information Systems and Management
Warwick Business School

Dimensionality Reduction

Dimensionality Reduction

- Part of data pre-processing
- Reduce the number of input variables, i.e. features, when dealing with high dimensional data
- Reduce the model complexity
- Improve performance and reduce computing time

Dimensionality Reduction

- Feature selection: to keep the most relevant variables from the original dataset
- Feature extraction: to get a smaller set of new variables that are "converted" from original variables, while retain the same information.

Feature selection

- Deselect features that are:
  - Variables with high missing value ratio
  - Variables with low variance
  - Variables that are highly correlated
  - Variables that are not important

Variable with Low Variance

- Low variance suggests the variable does not contribute much to the change of outcome.
  - Perform after scaling/standardization
  - Threshold to remove
- For dataframe, .var() returns variance for each column.
- sklearn also has VarianceThreshold() transformer that can remove features with low variance.

Variable with Low Variance

- sklearn.feature_selection module also has VarianceThreshold transformer that can remove features with low variance.
- Step 1: create the transformer VarianceThreshold

```
from sklearn.feature_selection import VarianceThreshold
vselector = VarianceThreshold(threshold=0.1) #default 0.0
```

- Step 2: apply the transformer with method fit_transform()

```
df_titan = vselector.fit_transform(df_titan)# if all numeric
```

Variables with High Correlation

- High correlation suggests variables may contribute similarly to the change of outcomes.
  - Some algorithms' performance can be heavily affected by high correlation.
  - Threshold to drop
- For dataframe, .corr() returns pair-wise correlation matrix.

Unimportant Variables

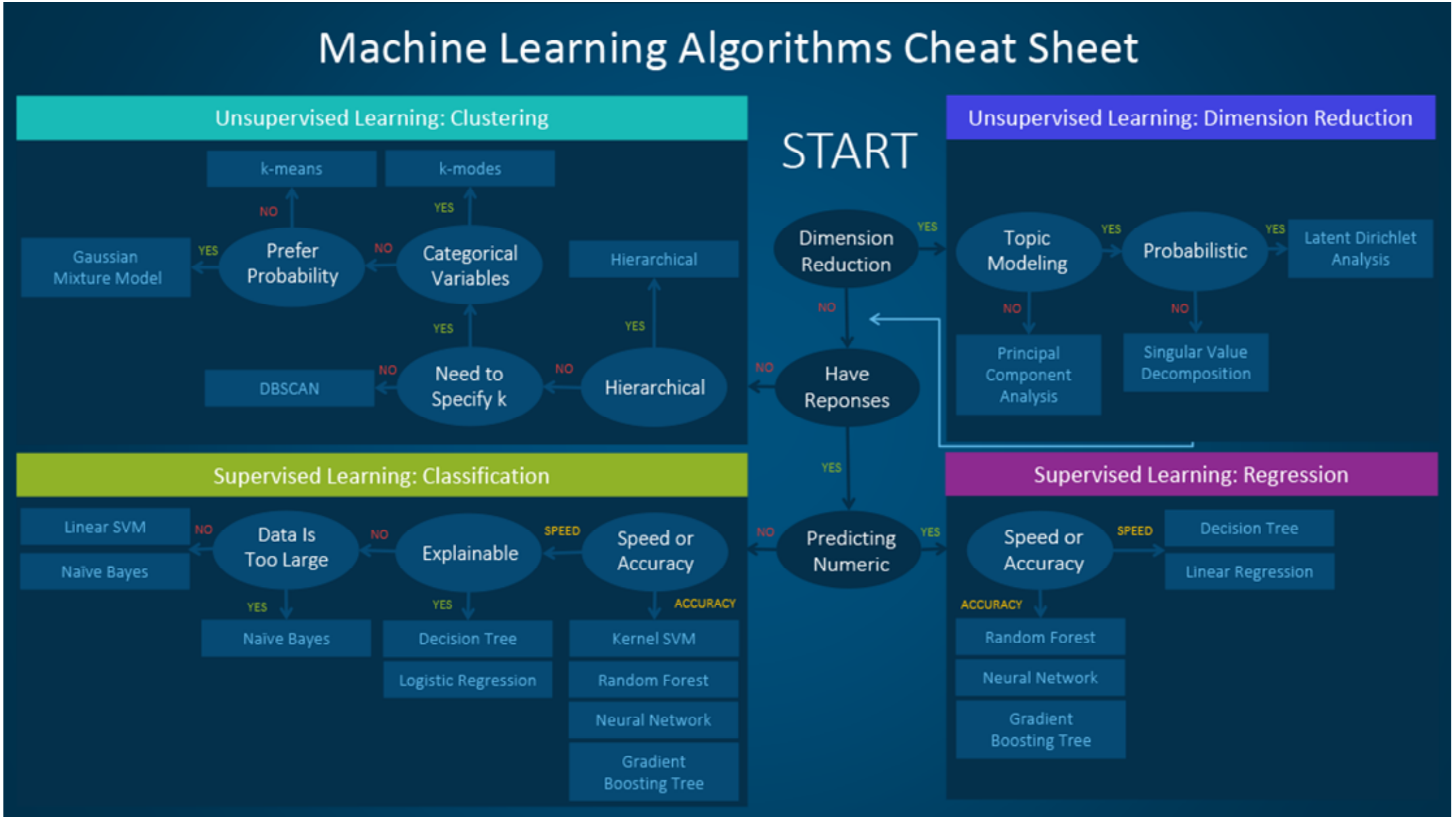
- Model-based techniques to eliminate unimportant variables.
- Any model produces feature importance (coef_ or feature_importances_): e.g. random forest
  - SelectFromModel from sklearn. feature_selection module can be used to create a transformer based on estimator.
- Backward Feature Elimination
  - RFE (recursive feature elimination) from sklearn. feature_selection module can be used to create a transformer based on estimator (coef_ or feature_importances_) .

Unimportant Variables

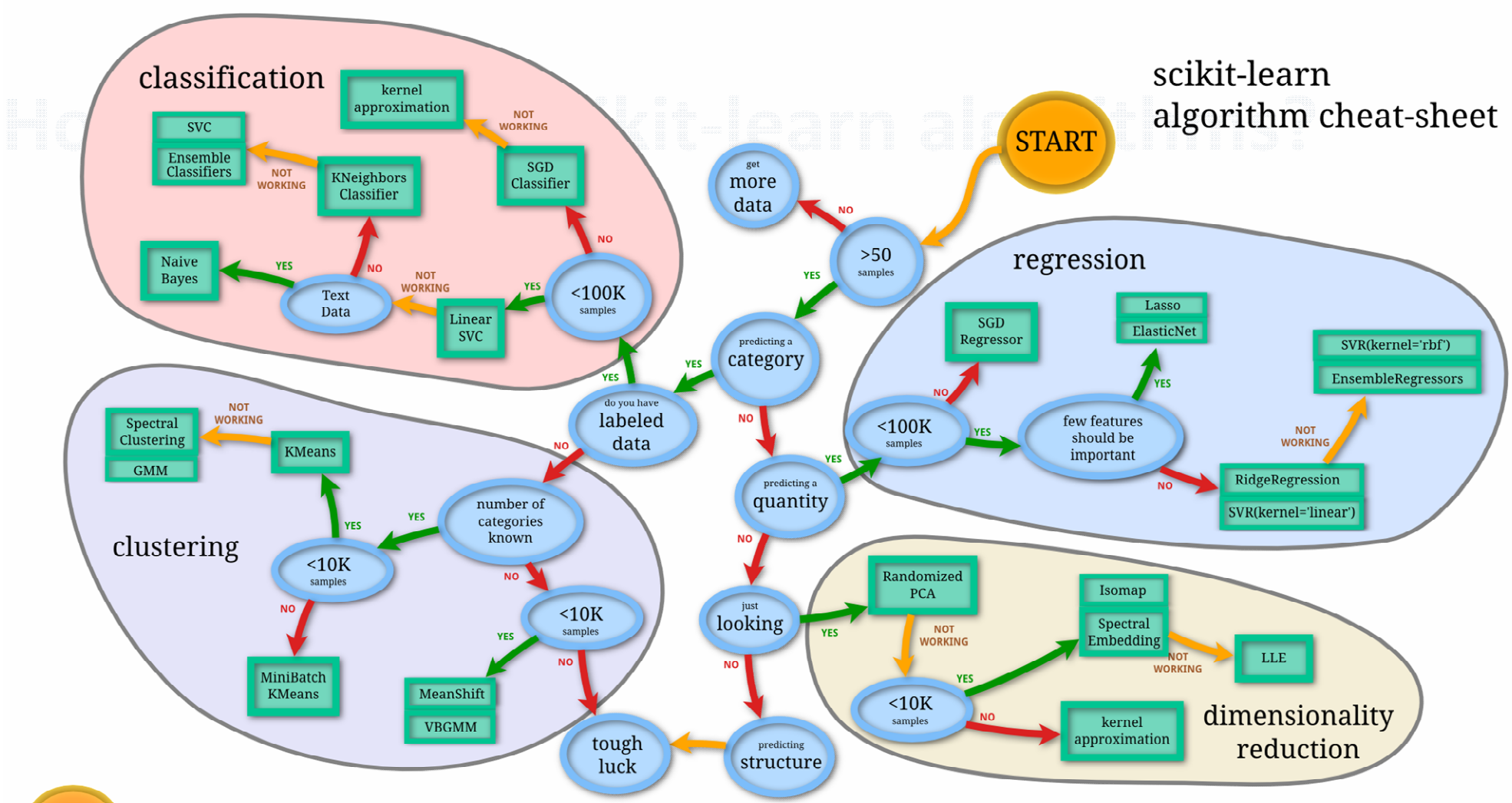
- Forward Feature Construction (0.24)
- SequentialFeatureSelector does not need coef_ feature_importances_

Feature Extraction

- Dimensionality reduction
- An unsupervised learning technique
- Often done after data cleaning and standardization/scaling



https://blogs.sas.com/content/subconsciousmusings/2017/04/12/machine-learning-algorithm-use/



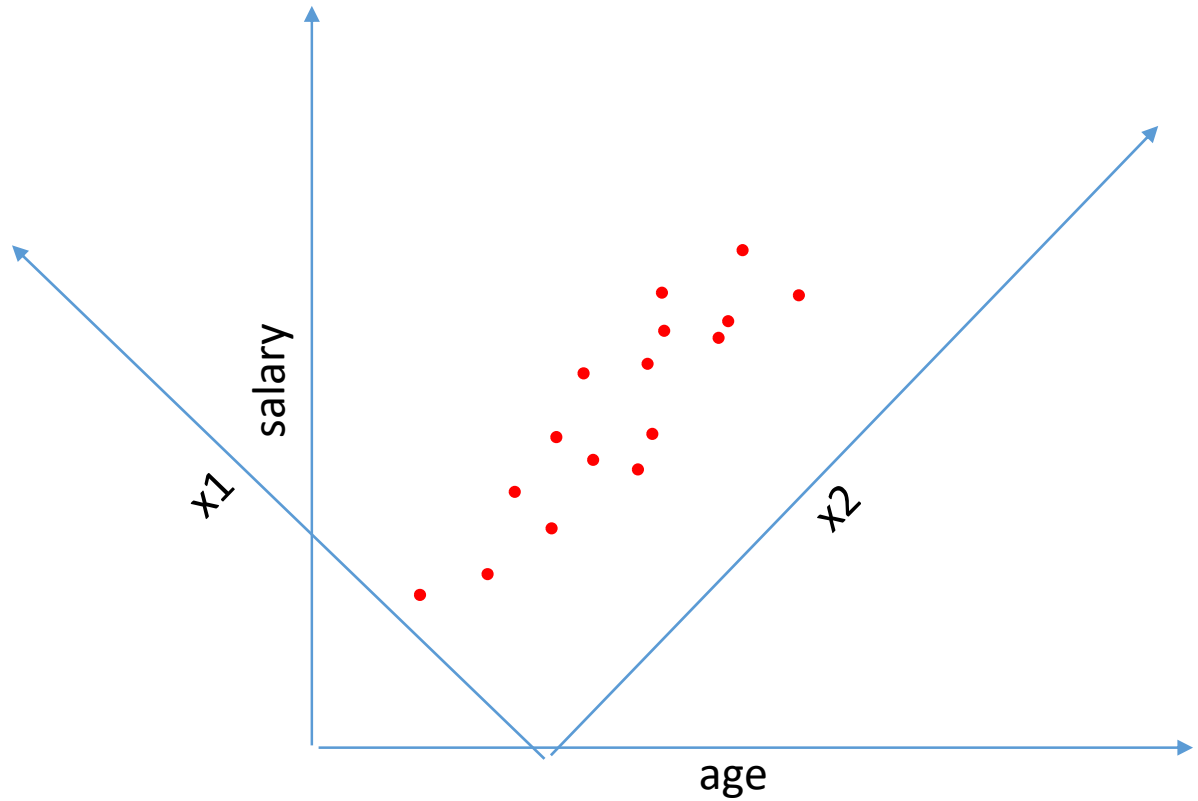
https://scikit-learn.org/stable/tutorial/machine_learning_map/

Feature Extraction

- linear methods
  - **Principal Component Analysis (PCA)**
  - Singular Value Decomposition (SVD)
- Nonlinear methods
  - **Kernel PCA**

Principal Component Analysis

- Most popular dimensionality reduction technique
- Works best with dense data
- Project original dataset into a different dimensional space with a new coordinate system
  - The principal component, i.e. new dimension, is decided by maximize the projected data.
  - Original features no longer exist and new features (meaningless) are generated.
  - Iteratively find the components and drop the least important ones if necessary.


salary
x1
x2
age

Principal Component Analysis

- The PCA class from sklearn.decomposition can be used to make PCA transformation.

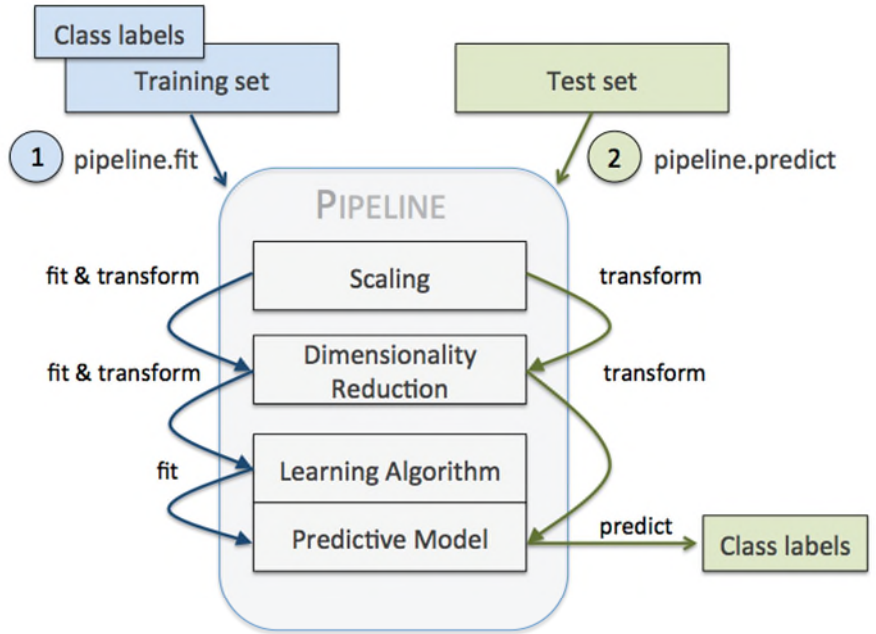
```
from sklearn.decomposition import PCA
pca = PCA(n_components=2)
X_train = pca.fit_transform(X_train)
```

Summary of data processing

- Feature engineering
- Feature selection
- Feature extraction(dimensionality reduction)
- Missing value
- Outlier
- Scaling/Standardization
- Categorical to numeric

Before or After Train-Test Splitting

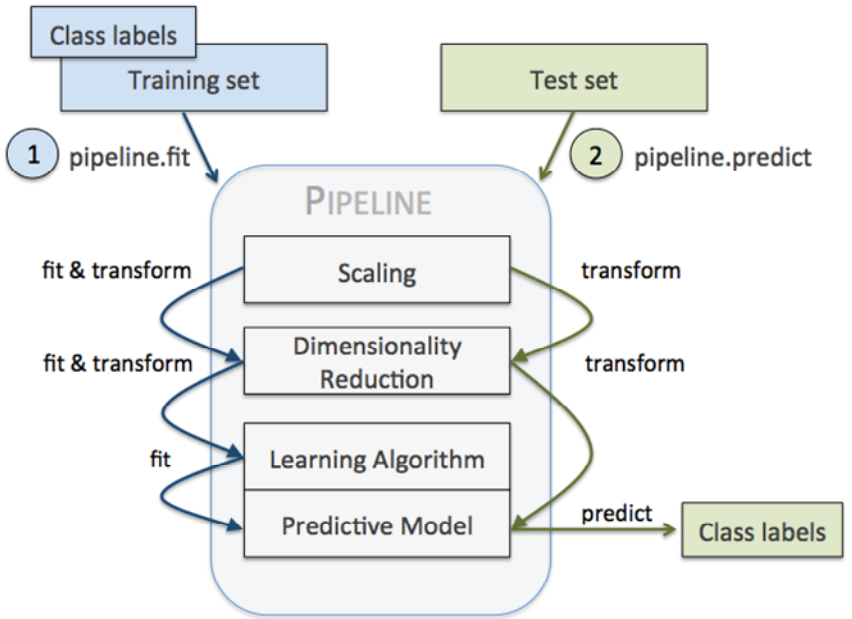
- Pre-processing should be done AFTER train-test splitting



Raschka, Sebastian. *Python machine learning*. Birmingham, UK: Packt Publishing, 2015. Print.

Machine Learning Pipeline in SKlearn

- Machine learning pipeline is a workflow to sequentially execute a series of tasks, such as imputation dimensionality reduction, and classification.
  - Easily reproduce the process
  - Automatic optimization;
  - Automatic selection.



Creating Pipeline

- Step 1: create the pipeline using Pipeline() class.

```
from sklearn.pipeline import Pipeline
pipe = Pipeline([('Scaler', MinMaxScaler()),
                 ('Enc', OneHotEncoder()),
                 ('clf', DecisionTreeClassifier())])
```

- Step 2: apply the pipeline and evaluate the model with method fit () and .score().

```
pipe = pipe.fit(X_train, Y_train)
pipe.score(X_test, Y_test)) #transform() or predict() can also be used depending on your goal.
```