Machine Learning in Python

Key takeaways

Types of Machine Learning

- **Supervised Learning:**
- Given independent data **X** and dependent data **y, the goal is to predict y from X.**
- If y takes categorical values → *Classification*
- If y takes numerical values → *Regression*

- **Unsupervised Learning:**
- Only **X** is given and the task is to find some subgroups/structure.

Methodology

- **Representation:**
- This is the hypothesis space. How do we represent our model?
- Choosing the ***hyperparameters*** determines the complexity/hypothesis space.
- Hyperparameters are user-defined inputs (e.g. Kernel function, max tree depth).

- **Evaluation:**
- We choose a criteria to say if a given model is good or not.
- In regression: mean squared error.
- In classification: accuracy, false positives (from confusion matrix).

- **Optimization:**
- The way we search for a good model on the hypothesis space.

Cross Validation

- **Holdout:**
- Leave a fraction of the data aside, train the model on the rest.
- **K-Fold:**
- Divide data in k parts, train in (k-1) of them and test in the other. Average the errors.
- Use k = 5 for quick prototyping, k = 10 for production/publication.
- **Leave one out:**
- Train the model on all the data except for one observation.

→ Best practice: Do Holdout *and* K-Fold.

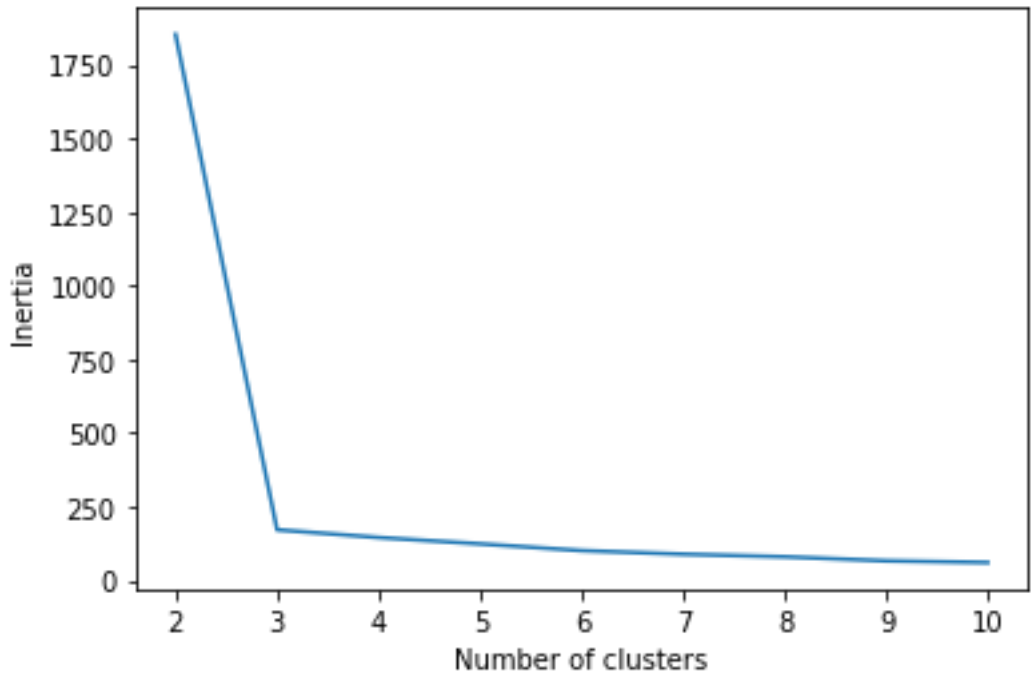
Learning Curve

- **High test error, low train error:**
- Overfitting. Simplify your model or add cross-validation.
- **Big gap in train and test error:**
- Problem with the data. Add more data, or check that the data in train and test set have similar statistical properties.
- **High value of train error:**
- Likely a bias problem, use a more complicated model (change hyperparameters).

Model Evaluation

Finding good clusters:

You can find the correct number of clusters using the *elbow method*.



More evaluation criteria

Classification

• Confusion matrix.

• ROC Curve

Regression

• Parity plot (prediction vs observed).

Preprocessing

• Recommended: numObservations > 10*numVariables.

• It is recommended to do normalization for linear/logistic regression and SVM. You can use *StandardScaler*, *MinMaxScaler*.

• Decision trees and random forests do not require normalization.

• For some data that has a lot of white noise, for instance, spectrometry data, keep in mind that normalization brings the white noise at the same scale of the signal. This means that your model might get trained on the noise, not on the spectra.

Algorithm	**When to use**	**When to avoid**
Linear / Logistic Regression	• Use as benchmark for both regression and classification.	• When the parity plot shows something different from a line at 45 degrees (regression).
Support Vector Machines	• When you have numerical features, can handle hundreds/thousands.	• If the data set has over 10,000 observations, training will be slow.
Decision Trees & Random Forests	• Variables with many categorical levels. • You can *'max_depth' and 'n_estimators'* (in Random Forests) to increase tree depth and solve nonlinear problems.	• High number of numerical variables (spectrometry data).
Neural Networks	• If you have more observations than variables (ideally much more than 10 observations per variable).	• When you need an explainable model. For instance, work that would be later checked by a regulatory agency.
K Nearest Neighbors	• To get an easy-to-explain model.	• Avoid if the train set is large. Since no coefficients are fitted, every new observation needs to be checked against all the observations in the training set.
K Means	• No ***y,*** you want to find structure in your set of observations. • Can be helpful as a preprocessing step: Do first k-means clustering in your data, and then use the cluster number as an additional variable.	