# Intro to Machine Learning

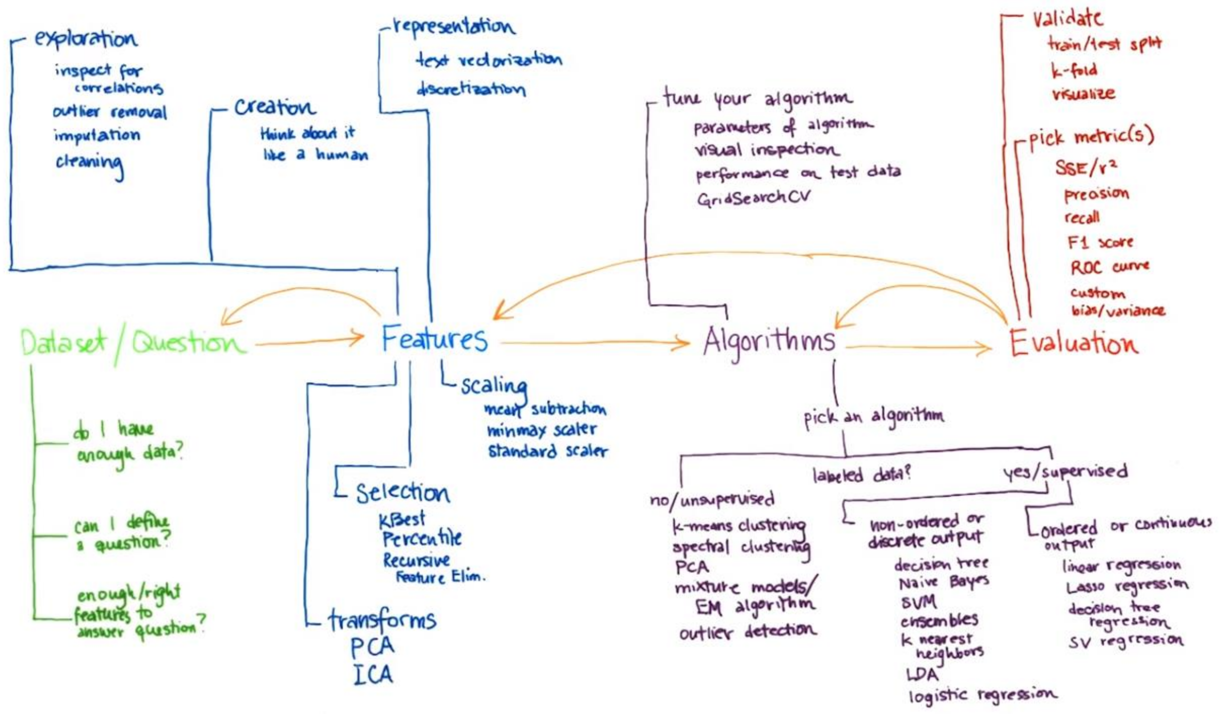
Main lecturer: Sebastian Thrun (Stanford, Google)

**This is a very brief overview of the course. For more detailed summary of similar concepts, see the courses summaries in *Intro to Artificial Intelligence* (also by Thrun), *Supervised Learning* and *Unsupervised Learning*.**

- Main code package: Python \ **SK-learn** – looks really good and convenient.
- Each section seems to end with a quite simple, medium-sized project to practice applying the new algorithm on some given data through SK-learn.
- **Naïve Bayes**:
  - Algorithm:
    - Define features xi.
    - For each class c learn f(xi|c) empirically.
    - Assign probability to new x using Bayes rule on $\{P_c \coloneqq \Pi_i f(x_i|c)\}_c$.
      - The last formula assumes independence of features, hence naïve.
  - Classic example: classifying text using bag-of-words approach.
- **SVM**: very shallow discussion.
  - The idea of maximum ***margin*** is explained.
  - The idea of non-linear ***separator*** using new fictive coordinates is demonstrated through the classic example of circle separator. It is mentioned without explaining that the new coordinates are yielded by some magical ***kernel***.
  - Two hyper-parameters (C & $\gamma$) which control smoothness & points-reach are briefly explained. Both seem to control the tradeoff between accuracy and avoiding overfitting.
    - The reach of a point determines how quick the influence of the point decays. Fast decay means that the separator will learn more locally (which overfits more).
  - It is demonstrated how to easily apply SVM with custom kernel & hyper-parameters using SK-learn.
- **Decision Trees**:
  - Basically a sequence of if-else's.
    - Equivalent to **step-function**.
  - Allows **non-linear** classification (and possibly regression) with **very simple basic blocks**.
  - A tree is learned by greedily splitting nodes that **maximize information gain** (i.e. cause largest reduce of entropy) or similar measures.
  - Trees have **high tendency to overfit**, hence it is important to use the hyperparameters (such as the minimum training samples required in a leaf that is considered for a split) to reduce overfit.
    - The **bias-variance tradeoff** of a learner (i.e. its sensitivity to the training data) was briefly discussed.
  - Many trees can be used together through the forest (and also ensemble?) algorithm.

- **Enron corpus**: corpus of emails of big company that crashed due to frauds.
    - Learn on various-sized training sets and compare validation errors to see dependence of performance on size of data is useful – in particular to see whether more data are needed.
- **Outliers**: can be ignored (if context justifies it) by training, detecting samples with large residual errors, remove them and re-train.
    - Another method (that was not mentioned) is detecting samples with "too large" influence on the learning, using ***Cook Distance***.
- **Clustering**: only k-means was discussed. Need to set number of clusters in advance. Retrying multiple times was suggested to deal with local minima.
- **Feature scaling**:
    - While many learning methods (e.g. linear regression, decision trees) aren't sensitive to feature scales, others are sensitive – sometimes unexpectedly (e.g. SVM and K-means for considering Euclidean distance in the input space; regularized linear regression). Hence, **scaling features by default is good practice**.
    - Most popular scaling methods are **min/max scaling** ($x \coloneqq (x - x_{min})/(x_{max} - x_{min})$) and **mean/var scaling** ($x \coloneqq (x - \mu)/\sigma$).
    - SK-lean inherently supports features scaling.
- **Text learning**: only **bag of words** approach was discussed (*count vectorizer* in SK-learn), along with basic pre-processing (***stopwords* deletion** and ***stemming***, both recommended through **NLTK**).
    - It was suggested to normalize words counting by the frequency of the word, so that rare words will have larger weights (at least in learning methods which are scale-sensitive).
- **Feature selection**:
    - Too many features may contain irrelevant information, cause overfitting, and/or slow down the learning.
    - Engineering smart features and choosing only the relevant ones are essential for the success of the training.
    - Initial feature selection can be done to filter out the less promising features, especially when feature space is huge (e.g. words counting space as in bag of words learning), and can be done **for example by greedily check for each feature its own potential as a single input**.
    - ***Regularization*** is also important, in particular one that actually zeroizes coefficients (such as ***Lasso*** which is supported by SK-learn).
- **PCA**: PCA + eigenfaces example.
- **Validation**: ***train & test sets*** (even validation set was not discussed); ***cross validation***.
- **Evaluation metrics**: mainly classification was discussed
    - ***Accuracy*** = rate of correct predictions – sensitive to:
        - Asymmetry in number of samples per class.
        - Asymmetry in cost of error in different classes.
    - ***Confusion matrix***: C[i,j] = P(predict j for instance of i).
        - ***Recall***(i) = TP(i)/(TP(i)+FN(i)) = P(correct | instance of i)
        - ***Precision***(i) = TP(i)/(TP(i)+FP(i)) = P(correct | i was predicted)
    - ***F-score*** was not discussed.

- **Summary**:


exploration
inspect for correlations
outlier removal
imputation
cleaning
creation
think about it like a human
representation
text vectorization
discretization
Dataset / Question
do I have enough data?
can I define a question?
enough/right features to answer question?
Features
scaling
mean subtraction
minmax scaler
standard scaler
selection
KBest
Percentile
Recursive Feature Elim.
transforms
PCA
ICA
tune your algorithm
parameters of algorithm
visual inspection
performance on test data
GridSearchCV
Algorithms
pick an algorithm
labeled data?
no/unsupervised
k-means clustering
spectral clustering
PCA
mixture models/ EM algorithm
outlier detection
yes/supervised
non-ordered or discrete output
decision tree
Naive Bayes
SVM
ensembles
k nearest neighbors
LDA
logistic regression
ordered or continuous output
linear regression
Lasso regression
decision tree regression
SV regression
Evaluation
validate
train/test split
k-fold
visualize
pick metric(s)
SSE/r²
precision
recall
F1 score
ROC curve
custom bias/variance