Models

Machine Learning with R

Basel R Bootcamp

October 2019

There is no free lunch

Theorem

Given a finite set V and a finite set S of real numbers, **assume that $f: V \to S$ is chosen at random** according to uniform distribution on the set S^V of all possible functions from V to S. For the problem of optimizing f over the set V, **then no algorithm performs better than blind search.**

Wolpert & Macready, 1997, No Free Lunch Theorems for Optimization

from christianfunnypictures.com

Know your problem

Bias-variance dilemma

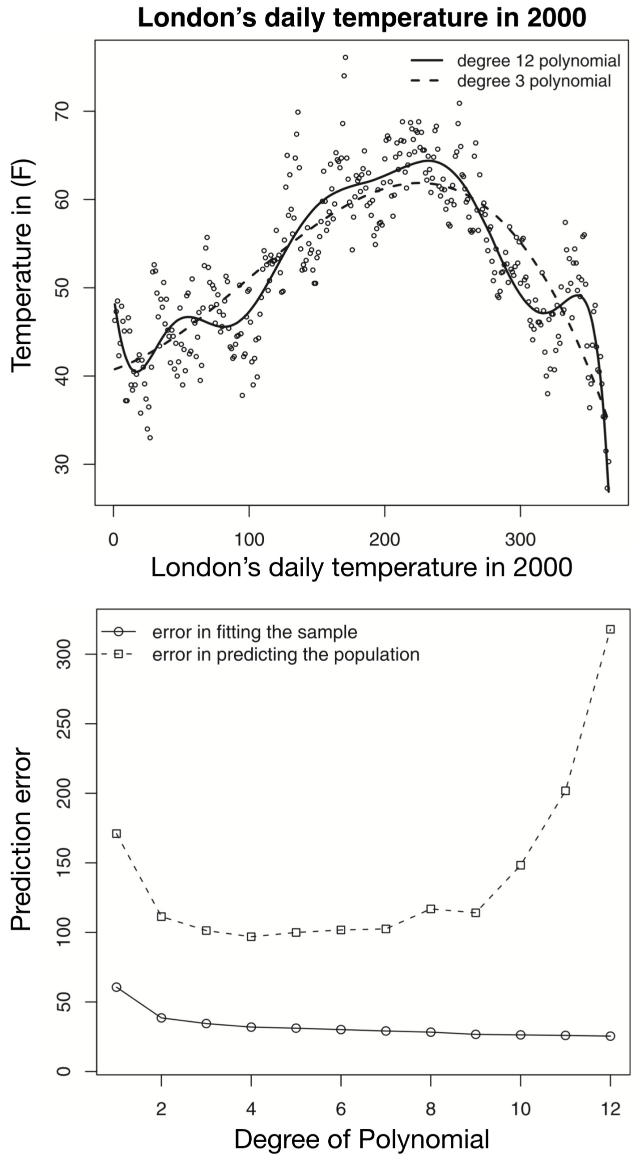
$$Error = Bias + Variance$$

Simply put...

Bias arises from strong **model assumptions** not being met by the environment.

Variance arises from high **model flexibility** fitting the noise in the data (i.e., overfitting).

→ **Make strong assumptions** (use simple models), if possible.

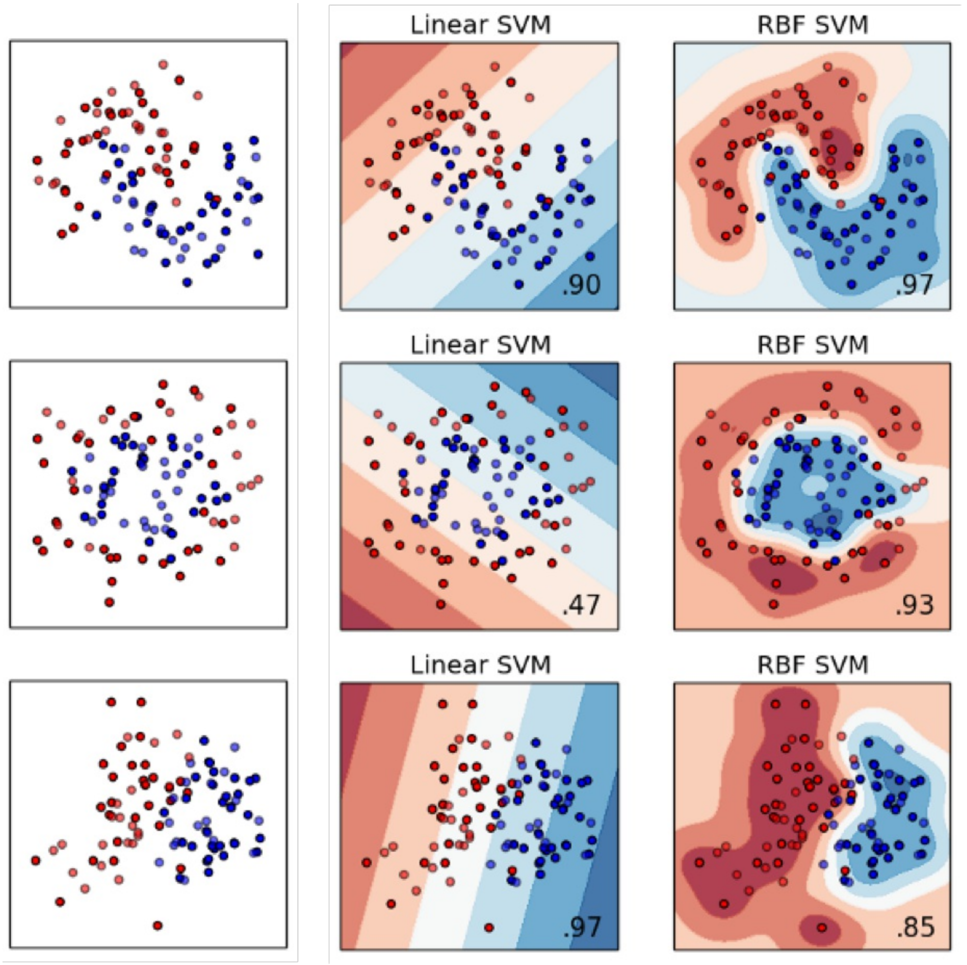


Linear or non-linear

One important model assumptions concerns linearity.

Linear models (`lm`, `glm`) make strong model assumptions. They are more often wrong, but also ceteris paribus **less prone to overfitting**.

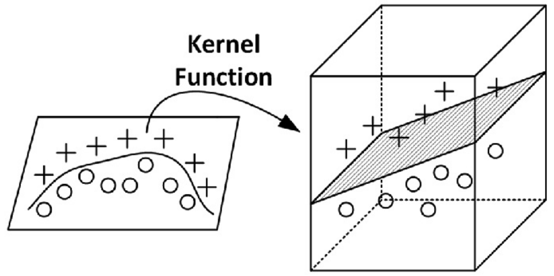
Non-linear Models (everything else) make weaker model assumptions, leaving the exact relationship (more) open. They are are closer to the truth, but also ceteris paribus **more prone to overfitting**.



from scikit-learn.org

Kernel trick

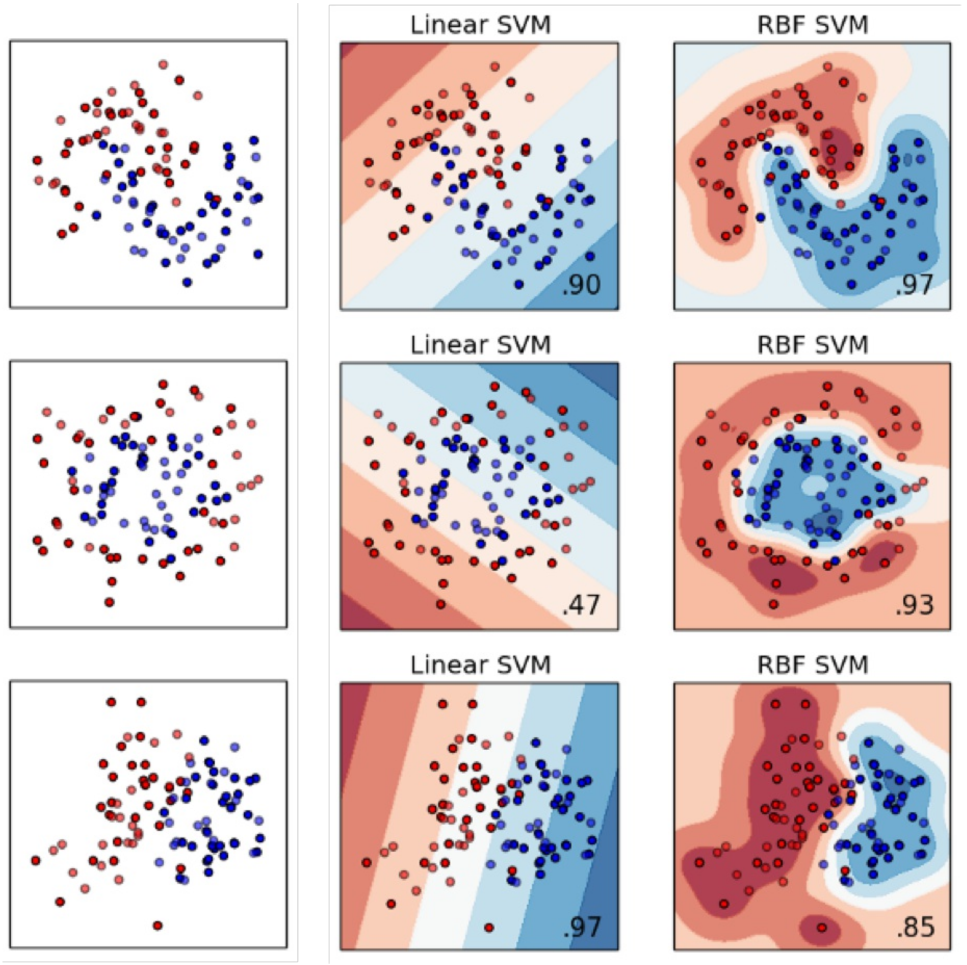
Transforms "input space" into new "feature space" to allows for object separation.



Used in **Support Vector Machines** (e.g., `method = "svmRadial"`) often using a **radial basis function** (rdf).

$$K(\mathbf{x}, \mathbf{x}') = \exp(-\gamma \|\mathbf{x} - \mathbf{x}'\|^2)$$

Kernels **re-represent objects** in terms of other objects!



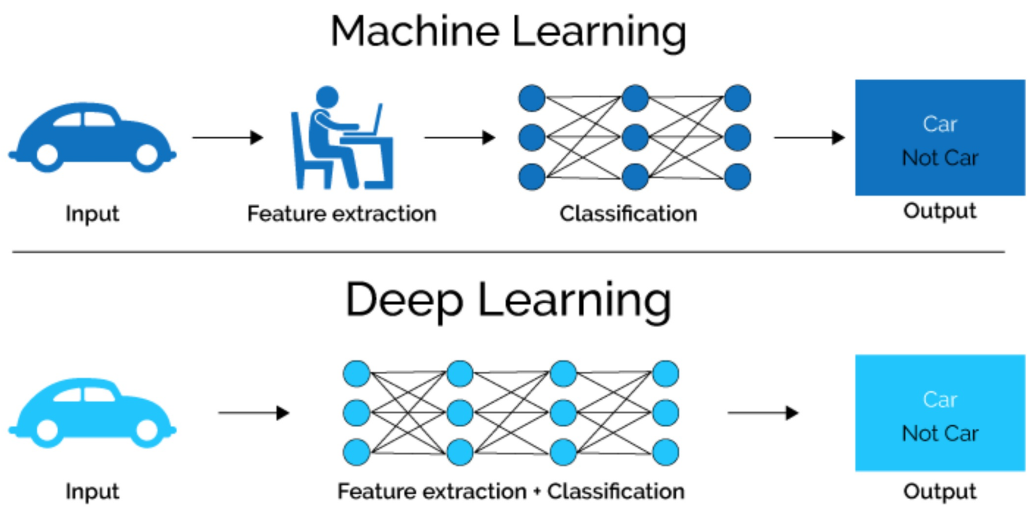
from scikit-learn.org

Automatic feature engineering

Deep learning aka neural networks and, especially, **convolutional neural networks**, excel because they generate their features.

Neural networks are not the focus of `caret` and this course. Powerful implementations based on **Google's Tensorflow** library are provided by `tensorflow`.

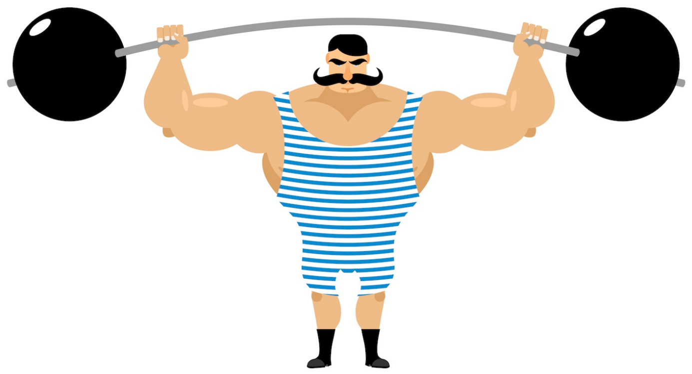
from towardsdatascience.com



from towardsdatascience.com

Robustness

To produce **robust predictions** that **suffer less from variance** ML models use a variety of **tricks**.



from istockphoto.com

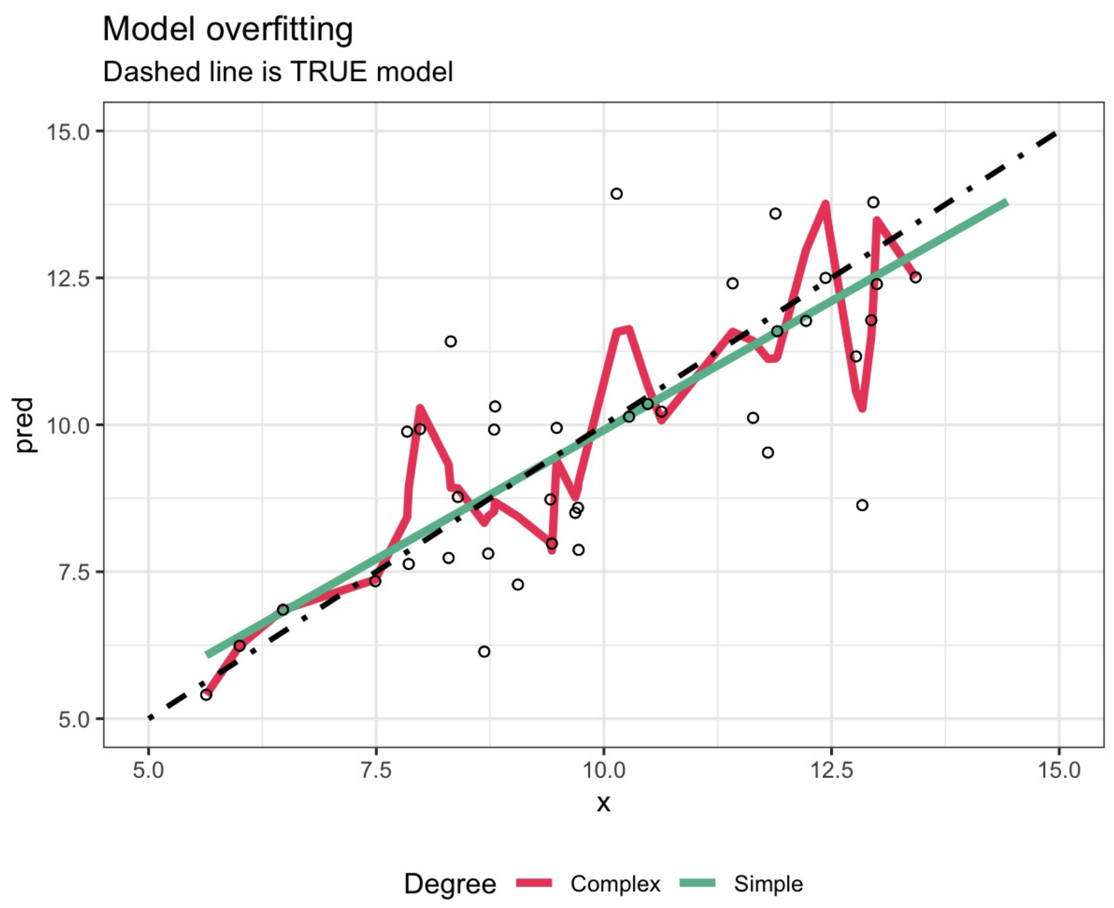
Approach	Implementation	Examples
Tolerance	Decrease error tolerance	`svmRadial`
Regularization	Penalize for complexity	`lasso`, `ridge`, `elasticnet`
Ensemble	Bagging	`treebag`, `randomGLM`, `randomForest`
Ensemble	Boosting	`adaboost`, `xgbTree`
Feature selection	Regularization	`lasso`
Feature selection	Importance	`random forest`

Regularization

Regularization is the process of adding model terms, usually **penalties for complexity**, in order to prevent overfitting (or solve a problem in the first place).

$$\text{Loss} = \text{Misfit} + \text{Penalty}$$

Name	Penalty	*caret*
AIC/BIC	$\|\|\beta\|\|_0$	-
Lasso	$\|\|\beta\|\|_1$	$method$ = glmnet
Ridge	$\|\|\beta\|\|_2$	$method$ = glmnet
Elastic Net	$\|\|\beta\|\|_2$	$method$ = glmnet



Bagging

Aggregate predictions from multiple fits to **resampled** data.

Especially beneficial for models that produce relatively unstable solutions, e.g., regression trees. `rpart` → `treebag`.

Algorithm

1 - **Resample** data (with replacement).

2 - **Fit** model to resampled data.

3 - **Average** predictions.

from wikipedia.org

Boosting

Bootsing **adaptively re-weights** samples based on performance.

`adaboost` and, newer, `xgbTree`, are some of the **best ML models out there**.

<u>Algorithm</u>

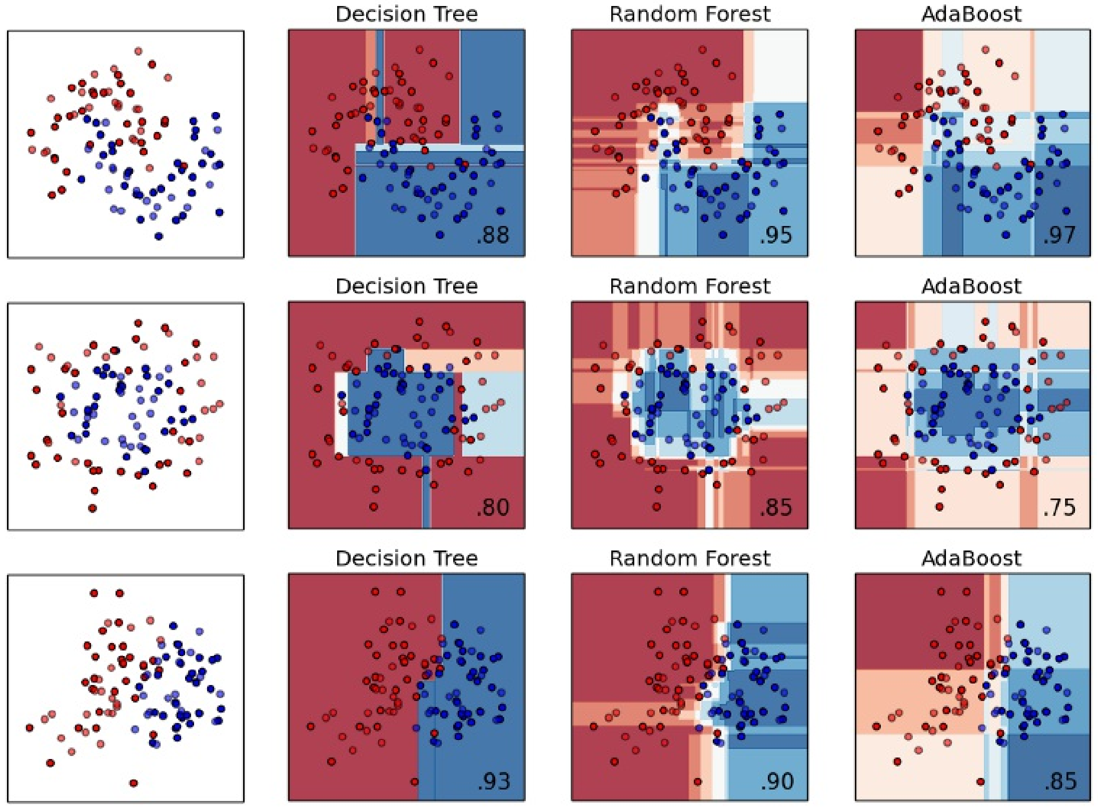
1 - Assign **equal weight** to all cases.

2 - **Fit** simple model.

3 - **Increase weight of misfit cases** by model misfit for next iteration.

4 - **Repeat**.

5 - **Average** predictions weighted by model misfit.



from scikit-learn.org

Automatic feature selection

Many models reduce complexity by automatically relying on a subset of good features.

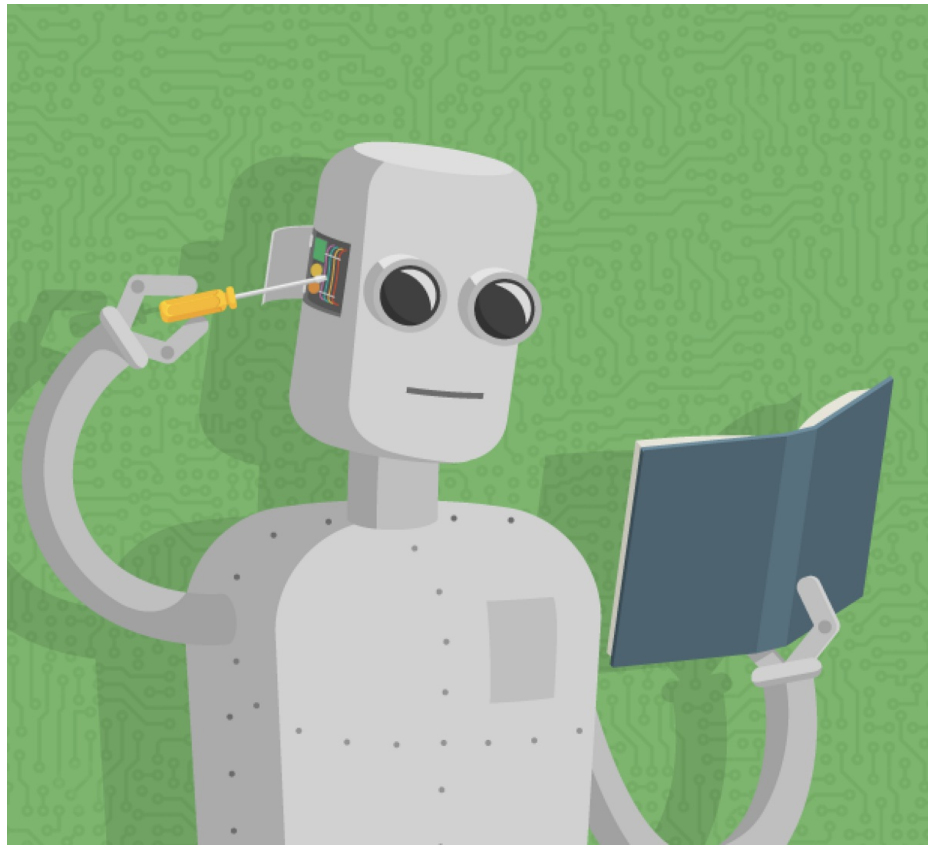
Two examples

LASSO

Regularization, in particular via `lasso`, frequently **estimates** `beta = 0` and, thus, essentially deselects that feature.

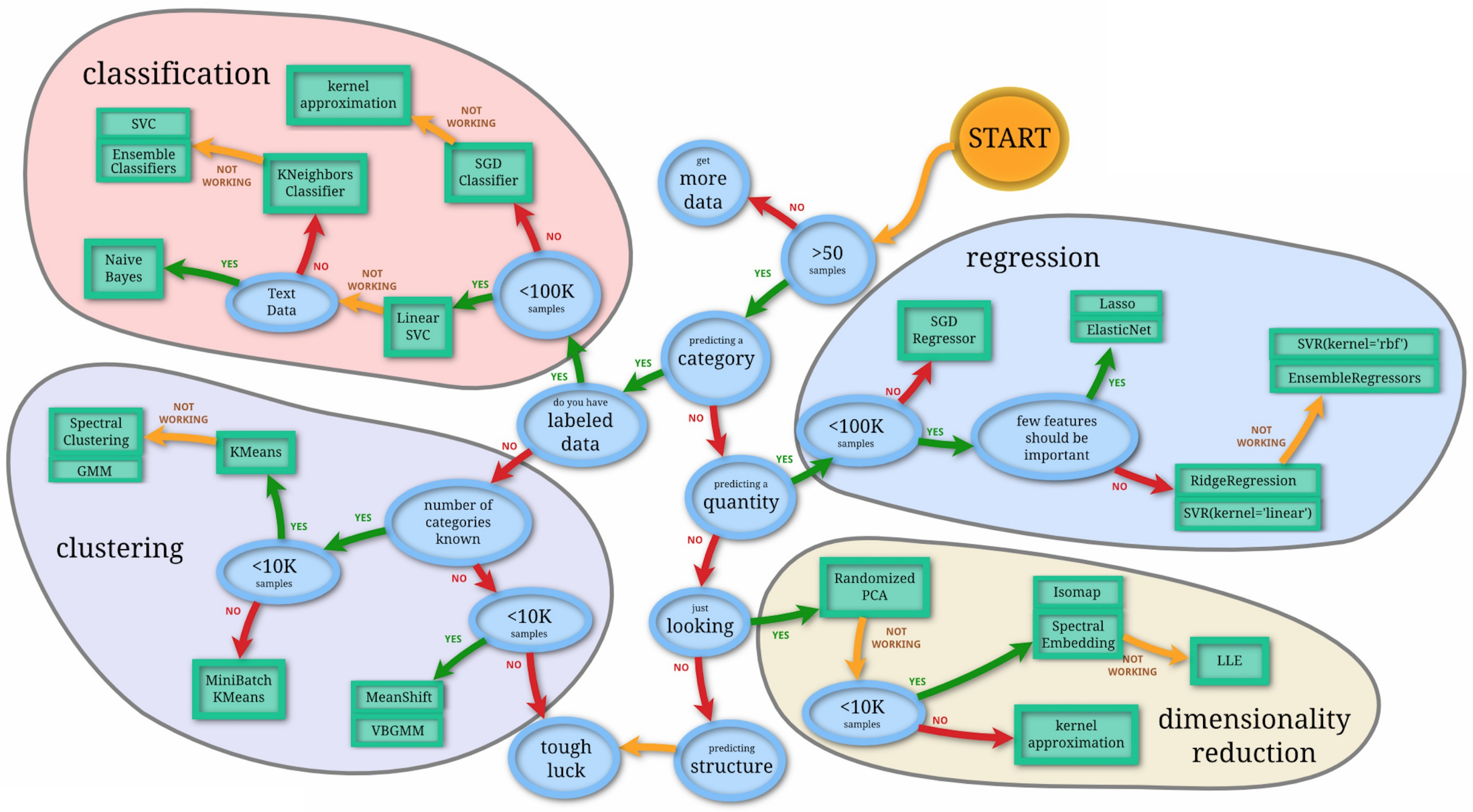
Random forests

As random forests select at any node the best of `mtry`-many randomly selected features, **unpredictive features may never come to action**. This is especially true for large `mtry`.



from medium.com

Some help in choosing models



from scikit-learn.org

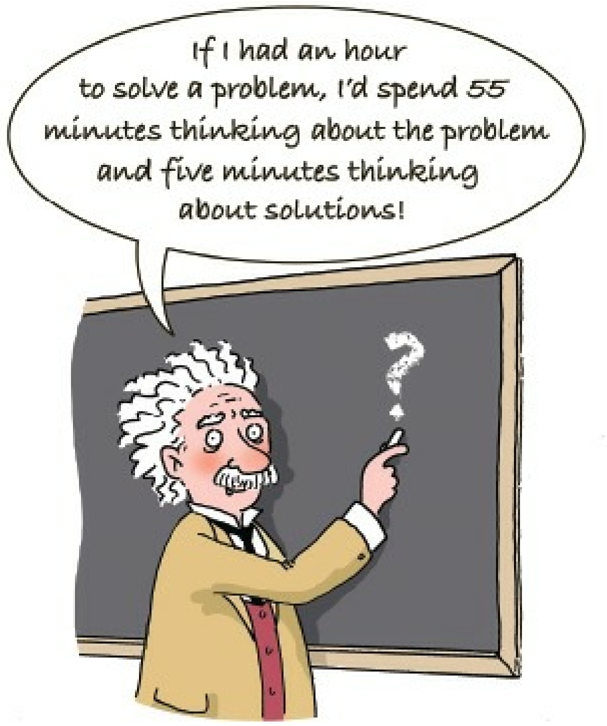
Remember

"…some machine learning projects succeed and some fail. What makes the difference? Easily the most important factor is the features used."

Pedro Domingos

"The algorithms we used are very standard for Kagglers. […] We spent most of our efforts in feature engineering. [...] We were also very careful to discard features likely to expose us to the risk of over-fitting our model."

Xavier Conort



from open.edu

Practical