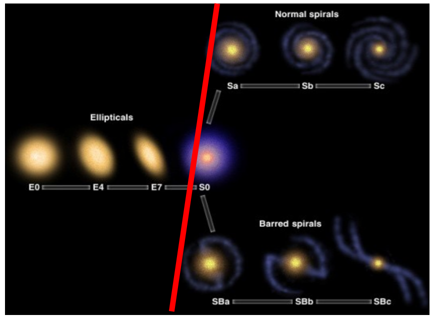
AUTOML: UNA BREVE INTRODUCCIÓN

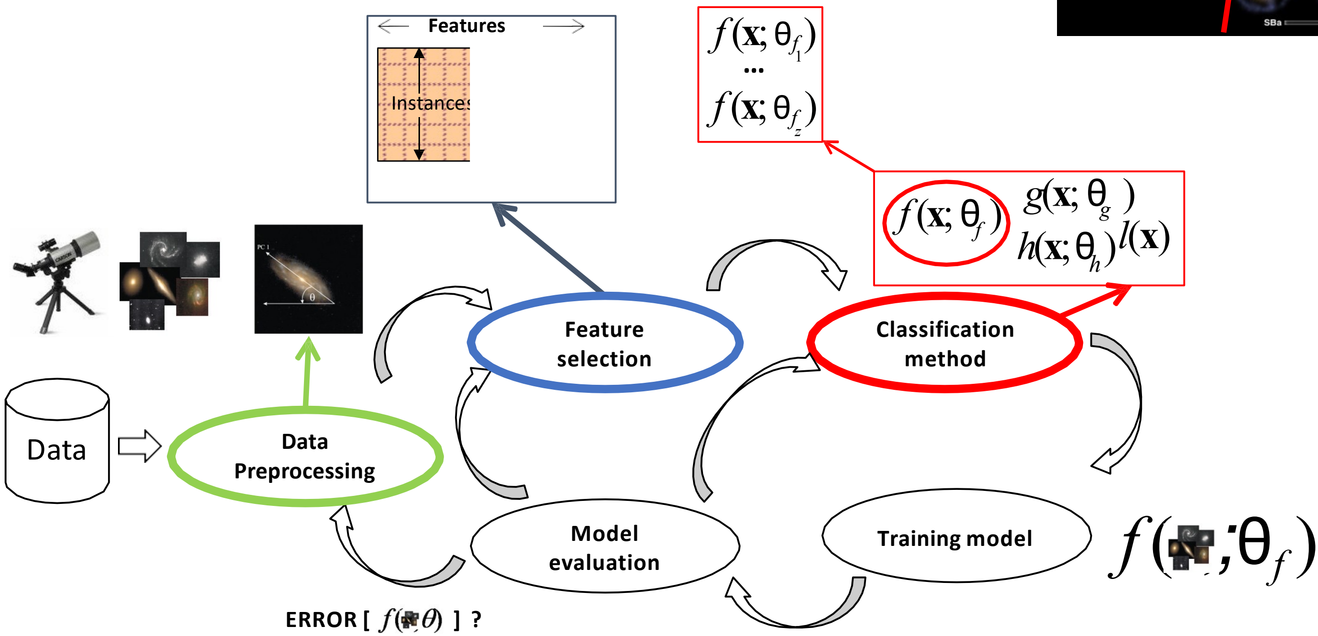
Minería de Datos: Aspectos Avanzados

Salvador García
salvagl@decsai.ugr.es

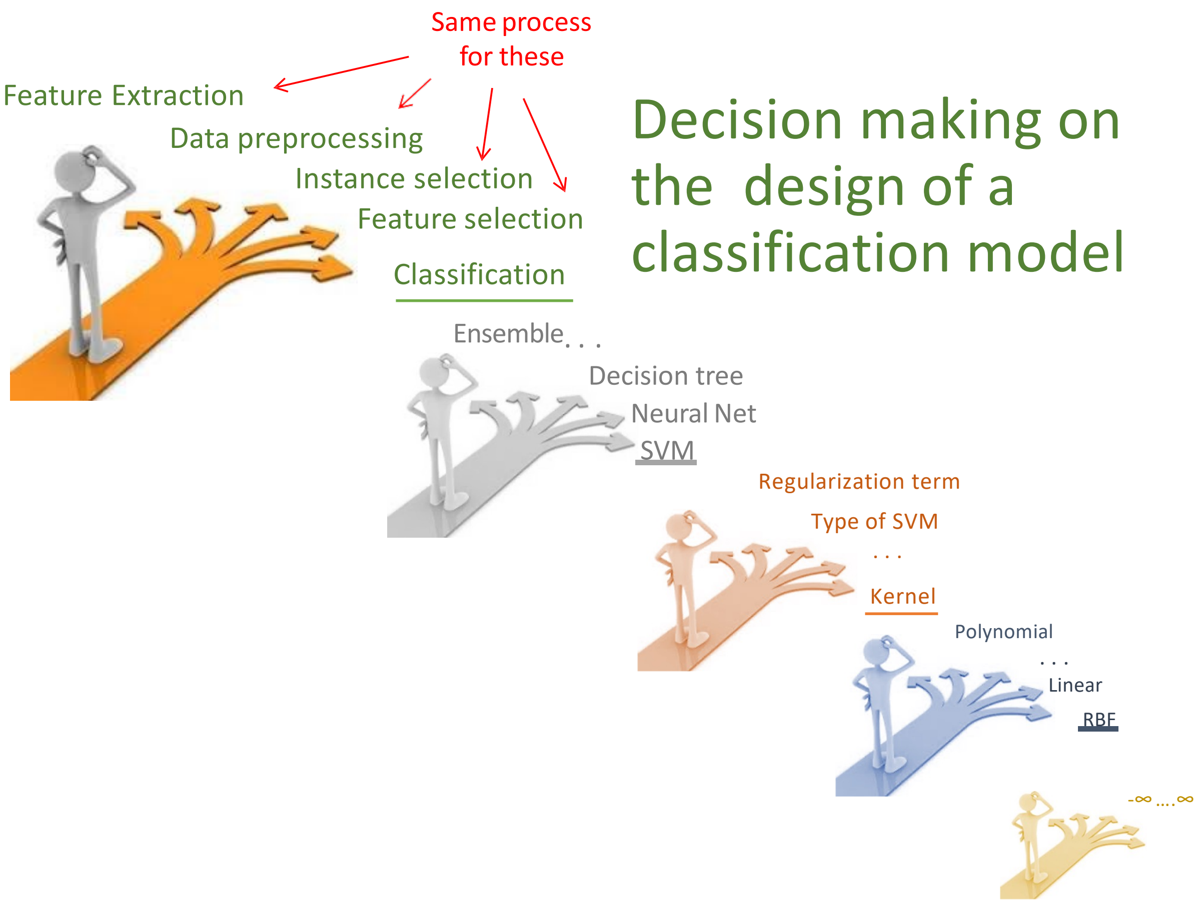
ML vs AutoML

Classification: Typical design process

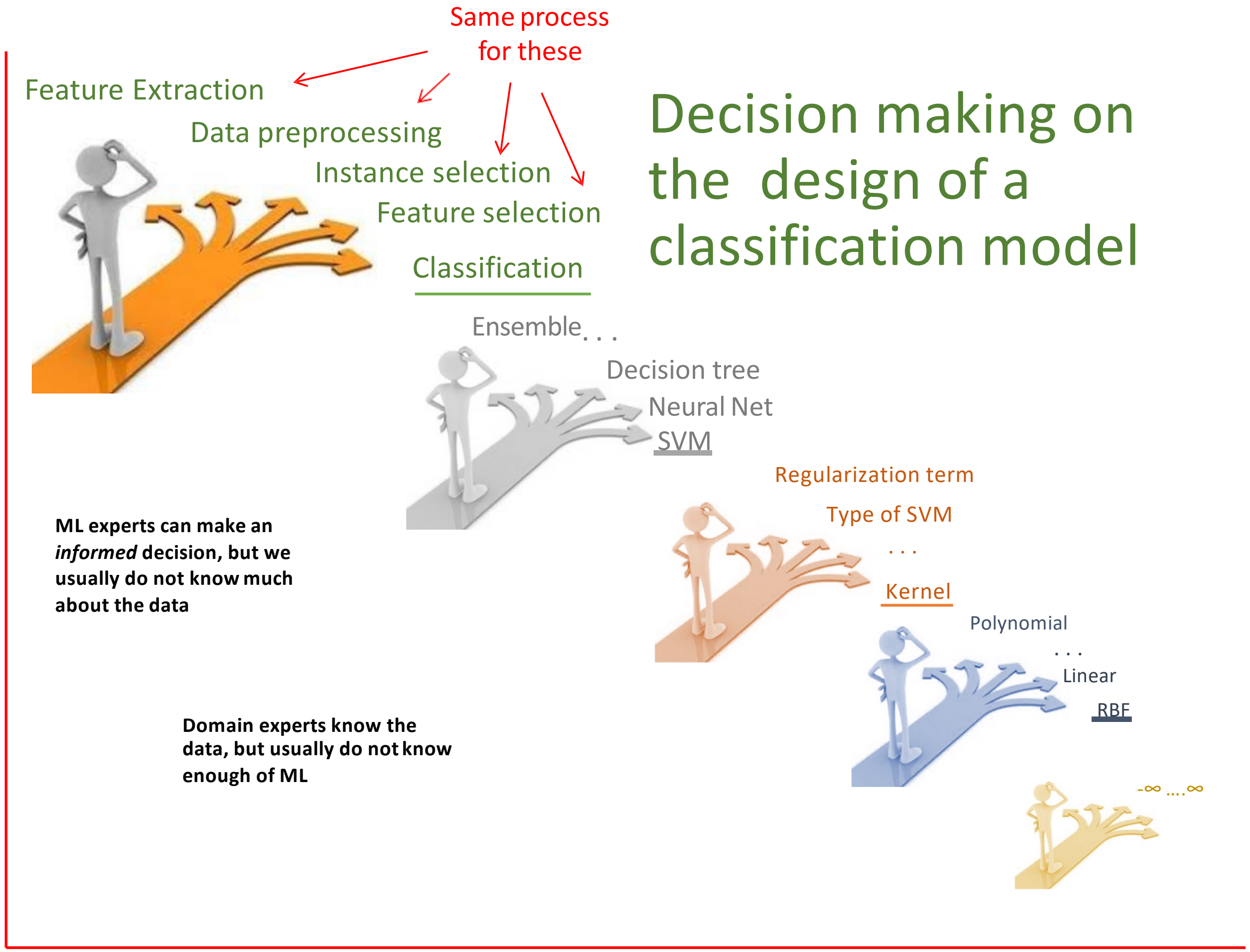




Decision making on the design of a classification model



Decision making on the design of a classification model

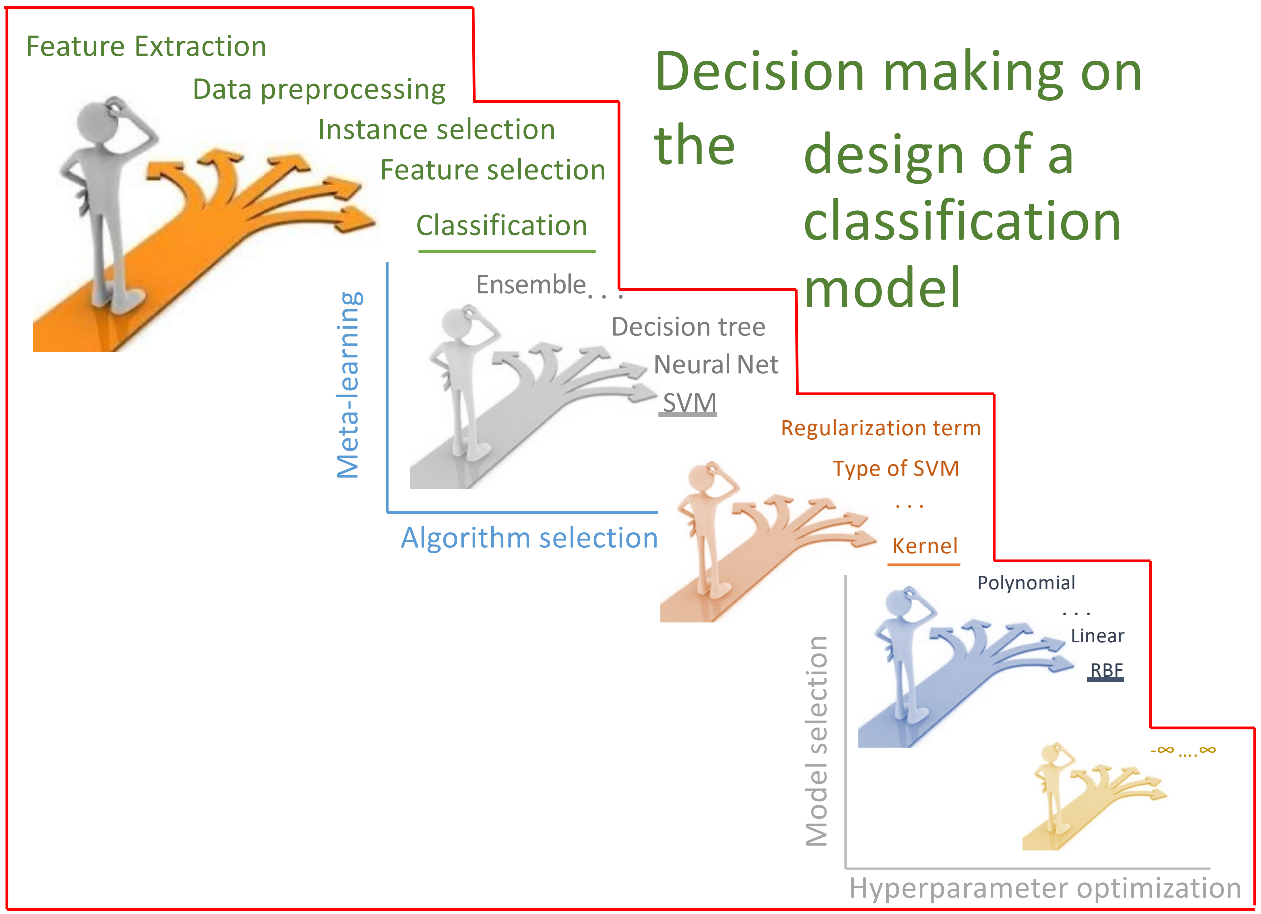


Some issues with the cycle of design

- Commonly, the above issues are fixed manually, relying on:
  - Domain expert's knowledge
  - Machine learning specialists' knowledge
  - Trial and error

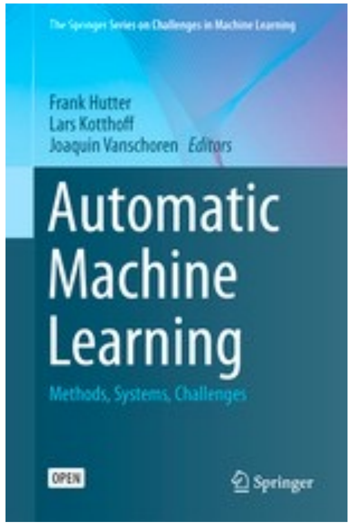
- The design/development of a pattern classification system relies on the knowledge and biases of humans, which may be risky, expensive and time consuming

- ***Automated solutions are available but only for particular processes (e.g., either feature selection, or classifier selection but not both)***


Decision making on the design of a classification model
Feature Extraction
Data preprocessing
Instance selection
Feature selection
Classification
Meta-learning
Ensemble. . .
Decision tree
Neural Net
SVM
Algorithm selection
Regularization term
Type of SVM
. . .
Kernel
Polynomial
. . .
Linear
RBF
Model selection
-∞ ....∞
Hyperparameter optimization

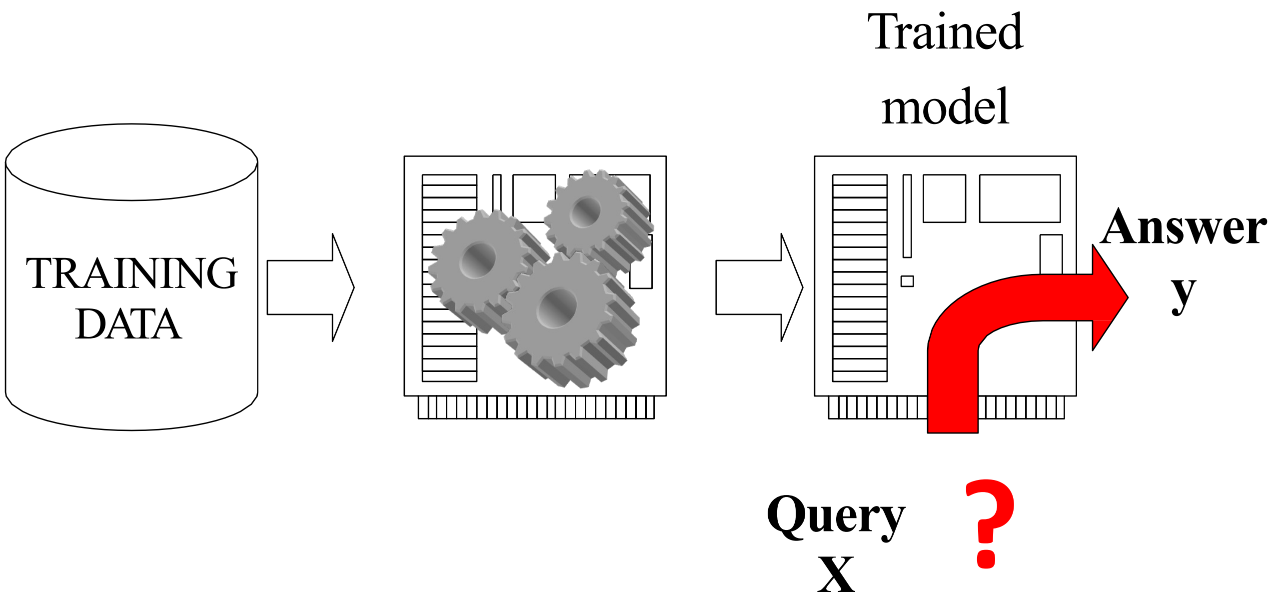
AutoML

- Automatic Machine Learning*
  - Research area that targets progressive automation of machine learning
  - Field of research focusing on the development of autonomous methods for solving a variety of machine learning problems

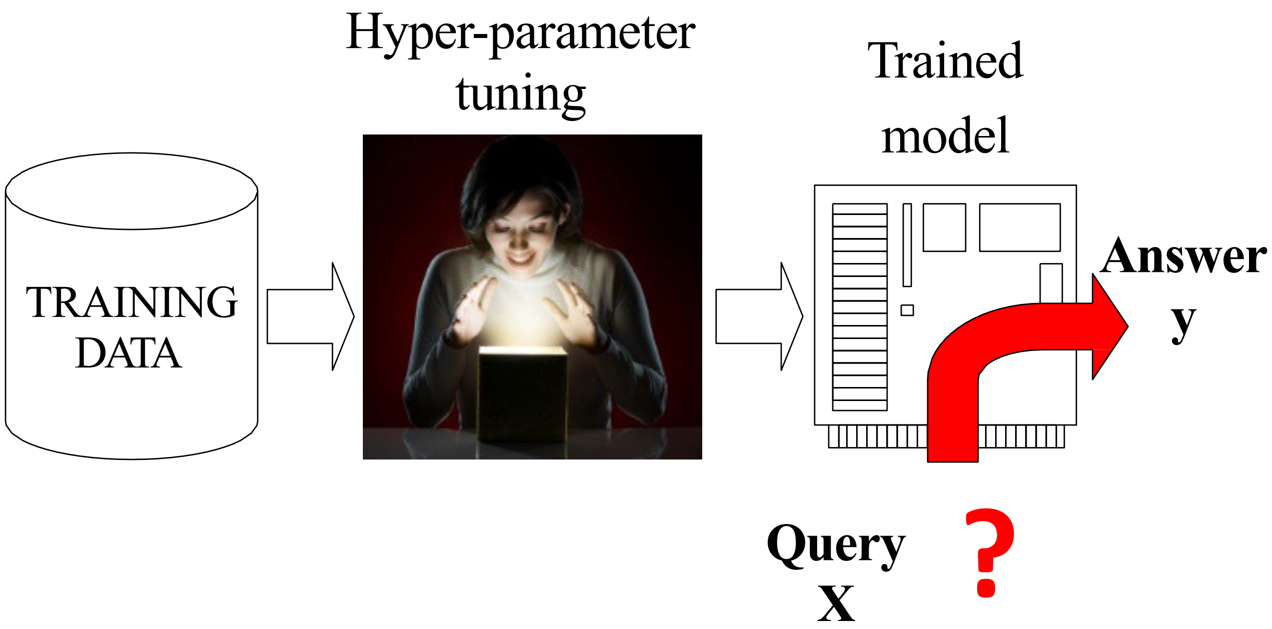


* We will focus on supervised learning

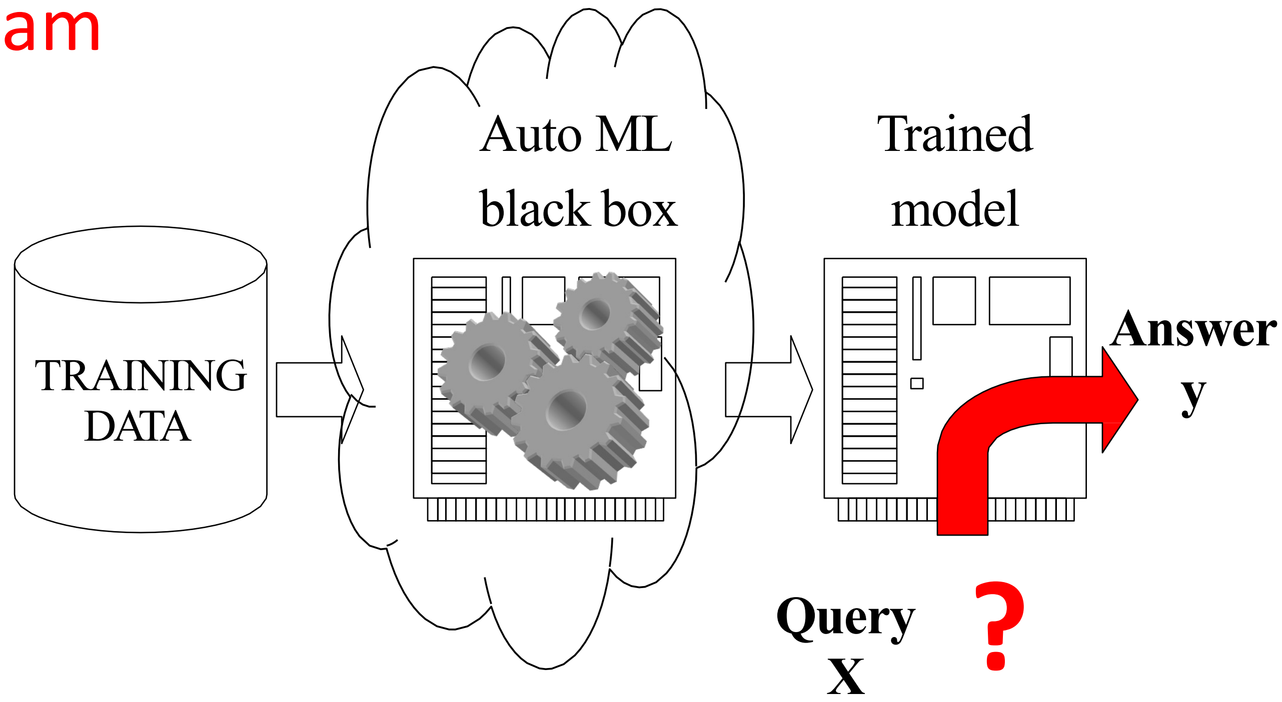
ML: The Problem



ML: The REALITY



AutoML: The Dream



Why AutoML?

- Relevance
  - Large amounts of data readily available everywhere
  - Lack of domain and/or ML experts who can advise/supervise the development of ML-based systems
  - Need to tune ML models

- Complexity
  - Huge and heterogeneous search space
  - Overfitting

AutoML notions

Informal, but intuitive definition

- *AutoML is* the *task of finding the model (f) that better generalizes in any possible dataset (T) with the less possible human intervention*
  - *f* can be the composition of multiple functions that may transform the input space, subsampling data, combining multiple predictors, etc.

  - where each of these models could be formed in turn by several other functions/models.

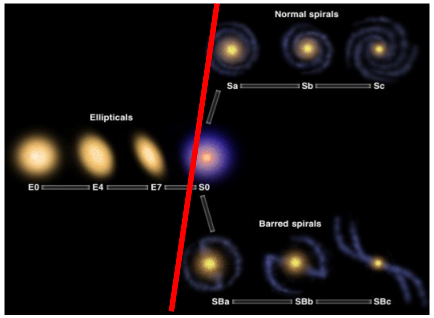
An inclusive notion of AutoML

- Three level categorization
  - Alpha-level - Search of estimators
  - Beta-level – Search of learning algorithms
  - Gamma-level – Search for meta-learning algorithms

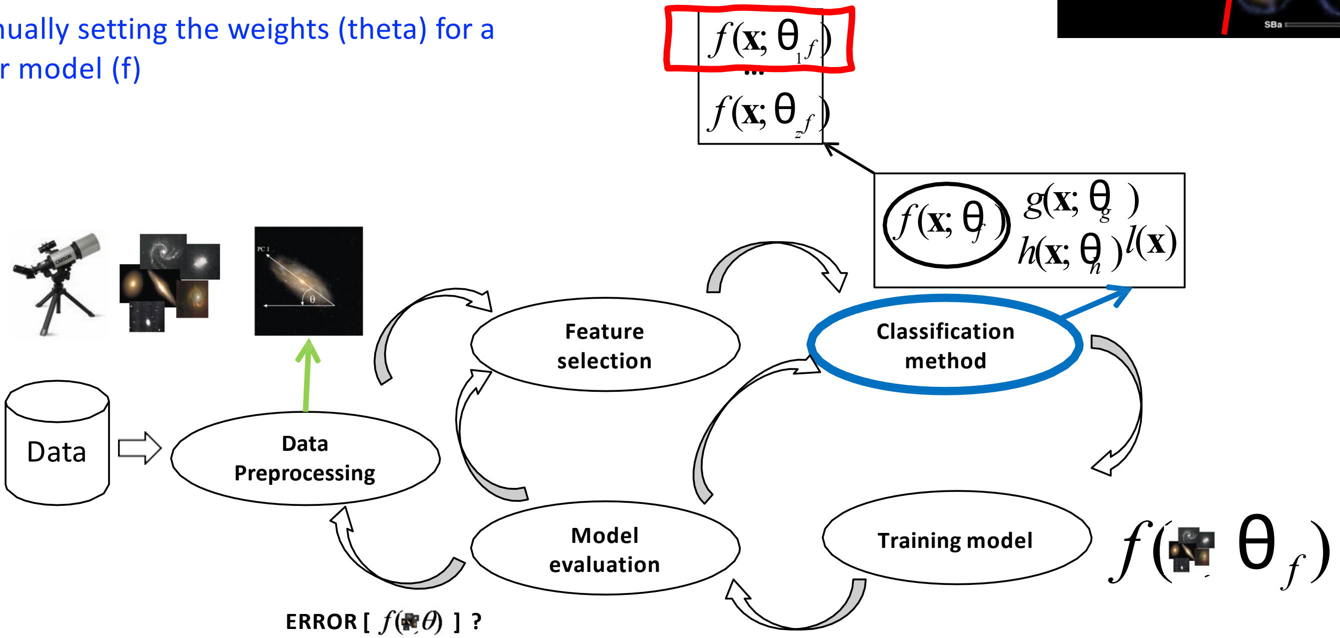
Level	Input	Output	Examples	Encoded by
α-level	sample/example (e.g. an image)	prediction of label (e.g. 'dog' or 'cat')	heuristically hard-coded classifier or already trained classifier	parameters, hyperparameters (if any) and meta-parameters (if any)
β-level	task/dataset (e.g. MNIST [20], CIFAR-10 [19])	α-level algorithm	learning algorithms (e.g. SVM [6], CNN [20]); HPO algorithms (e.g. grid search cross-validation, SMAC [1], NAS [34])	hyperparameters and meta-parameters (if any)
γ-level	meta-dataset (e.g. OpenML [29])	β-level algorithm	meta-learning algorithms (e.g. meta-learning part in Auto-sklearn [11])	meta-parameters

Liu Zhengying, Zhen Xu, Meysam Madadi, Julio Jacques Junior, Sergio Escalera, Shangeth Rajaa, and Isabelle Guyon. **Overview and unifying conceptualization of automated machine learning.** In Proc. of Automating Data Science Workshop @ECML-PKDD, 2019.

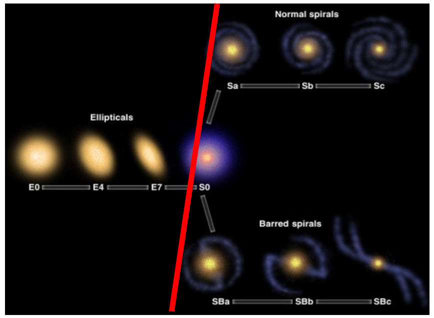
Alpha level



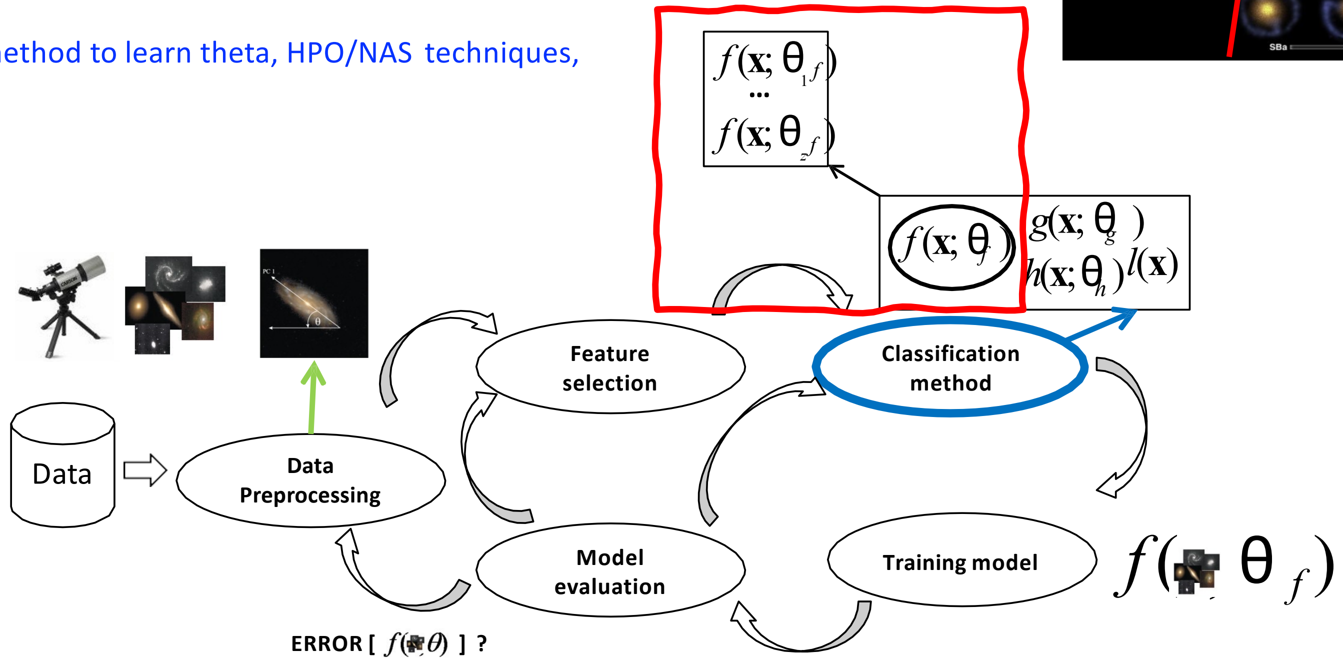
e.g., Manually setting the weights (theta) for a particular model (f)



Beta level

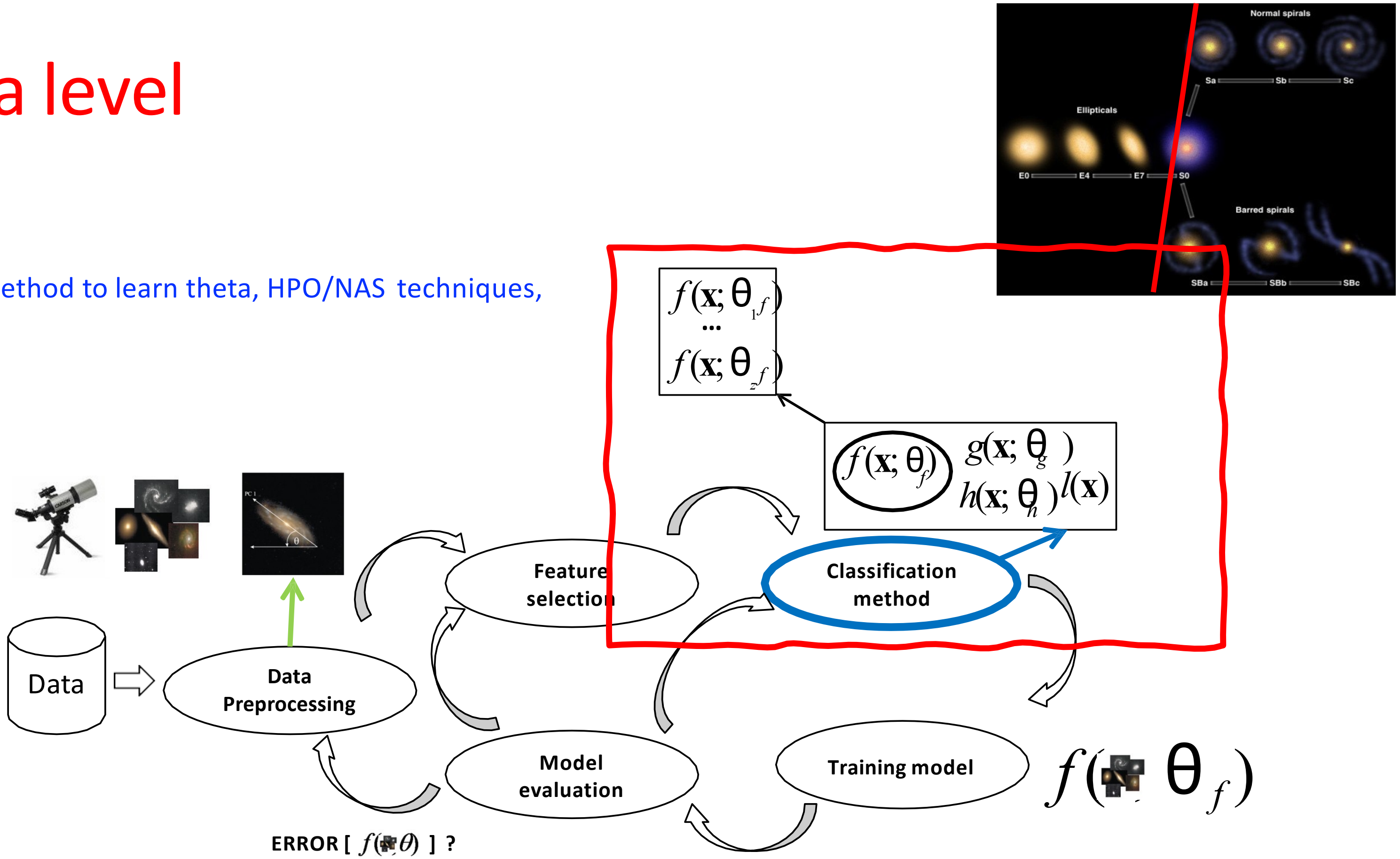


e.g., A method to learn theta, HPO/NAS techniques,



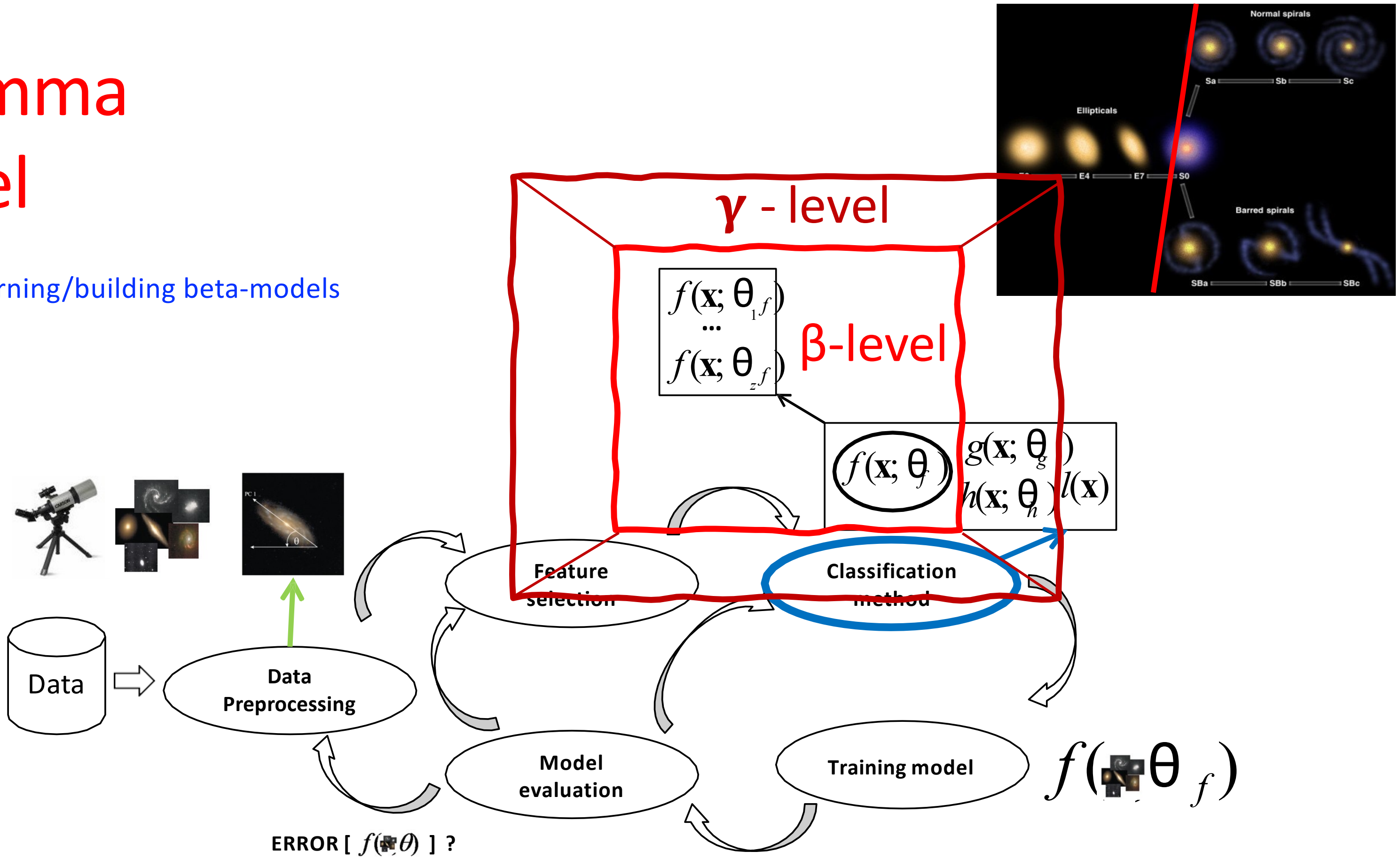
Beta level

e.g., A method to learn theta, HPO/NAS techniques,



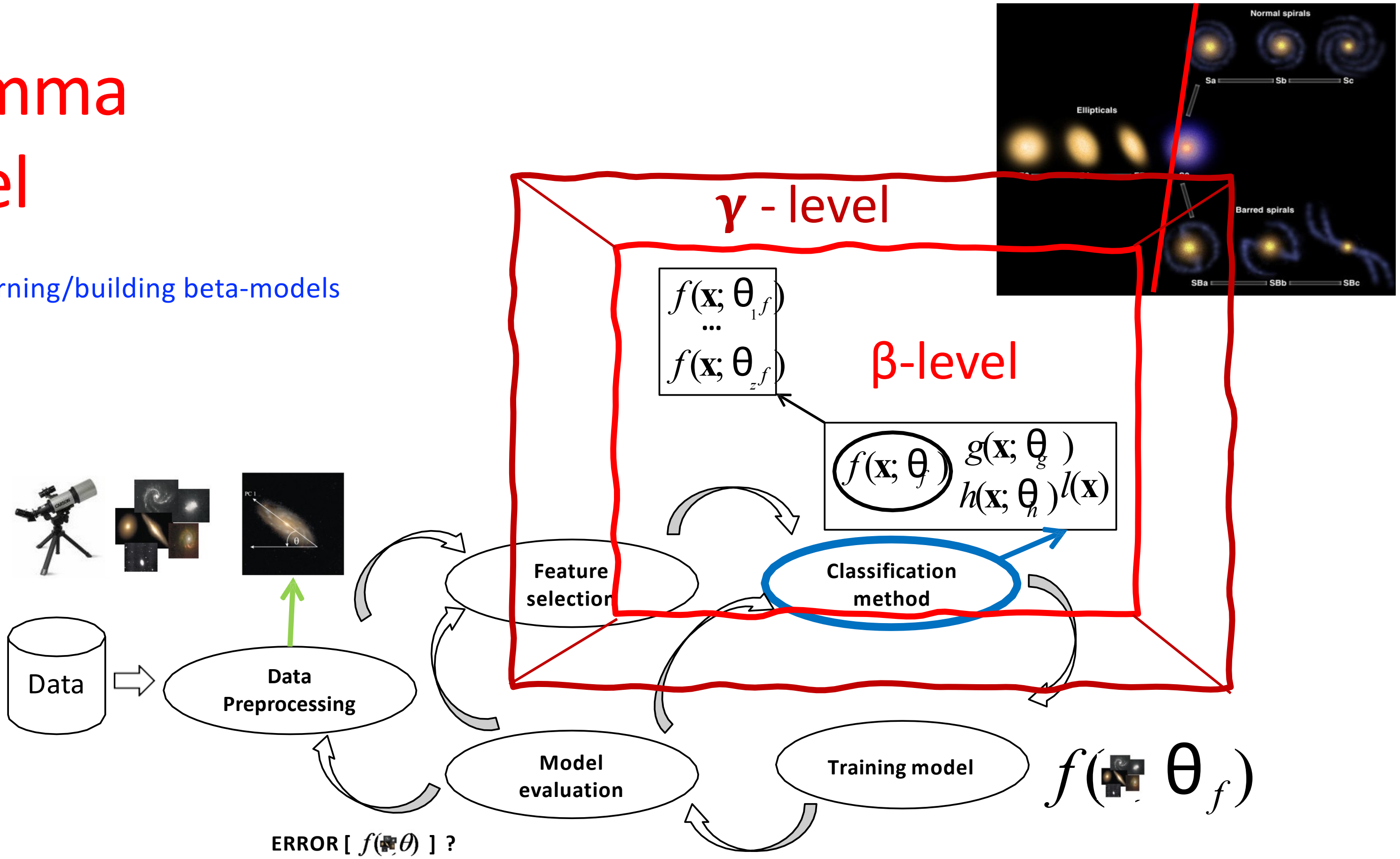
Gamma level

e.g., Learning/building beta-models



Gamma level

e.g., Learning/building beta-models



The three waves of AutoML

Three waves of AutoML

2006-2011

Full model selection, first surrogates, ensembles of solutions

2012 - 2017

SMBO, AutoSKlearn, AutoWEKA, TPOT

2018 - ?

Neural Architecture Search, RL

Renascence of Meta Learning

Few shot learning, etc.

First wave (2006 – 2011)

- Meta learning (early efforts)
  - Algorithm selection, classifier recommendation
- Full model (pipeline) selection vs HPO
- Ensembles of partial solutions
- Surrogate models

PSMS : PSO for full model selection

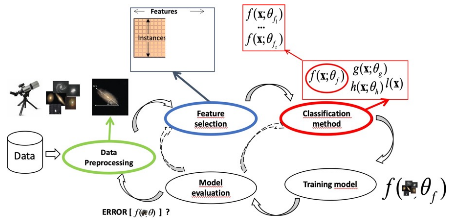
- Codification of solutions as real valued vectors

Hyperparameters for the selected methods

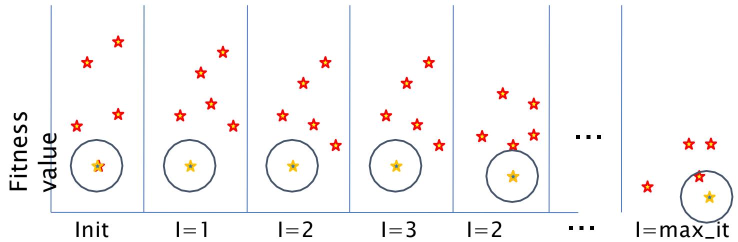
Preprocessing before feature selection?

$$\mathbf{x}_i = \left\langle x_{\mathrm{i,pre}}, y_{\mathrm{i,1...Npre}}, x_{\mathrm{i,fs}}, y_{\mathrm{i,1...Nfs}}, x_{\mathrm{i,sel}}, x_{\mathrm{i,class}}, y_{\mathrm{i,1...Nclass}} \right\rangle$$

Choice of methods for preprocessing, feature selection and classification



- Using PSO straightforwardly



PSMS : PSO for full model selection

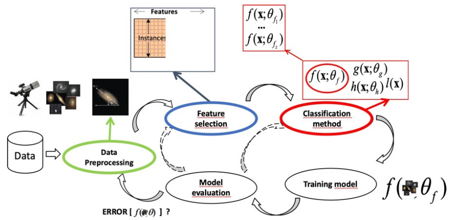
- Codification of solutions as real valued vectors

Hyperparameters for the selected methods

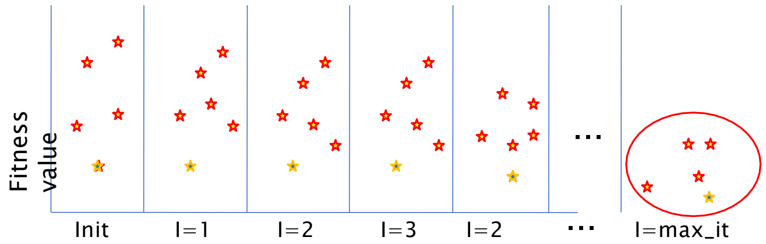
Preprocessing before feature selection?

$$\mathbf{x}_i = \langle \boxed{x_{\mathrm{i,pre}}}, \boxed{y_{\mathrm{i,1...Npre}}}, \boxed{x_{\mathrm{i,fs}}}, \boxed{y_{\mathrm{i,1...Nfs}}}, \boxed{x_{\mathrm{i,sel}}}, \boxed{x_{\mathrm{i,class}}}, \boxed{y_{\mathrm{i,1...Nclass}}} \rangle$$

Choice of methods for preprocessing, feature selection and classification



- Using PSO straightforwardly



Main findings from the early years

- Overfitting avoidance mechanisms
- Heterogeneous codification of solutions
- “Straightforward” global optimization approaches
- Data subsampling for efficient estimation of the objective function
- Assembling solutions
- Template-based initialization

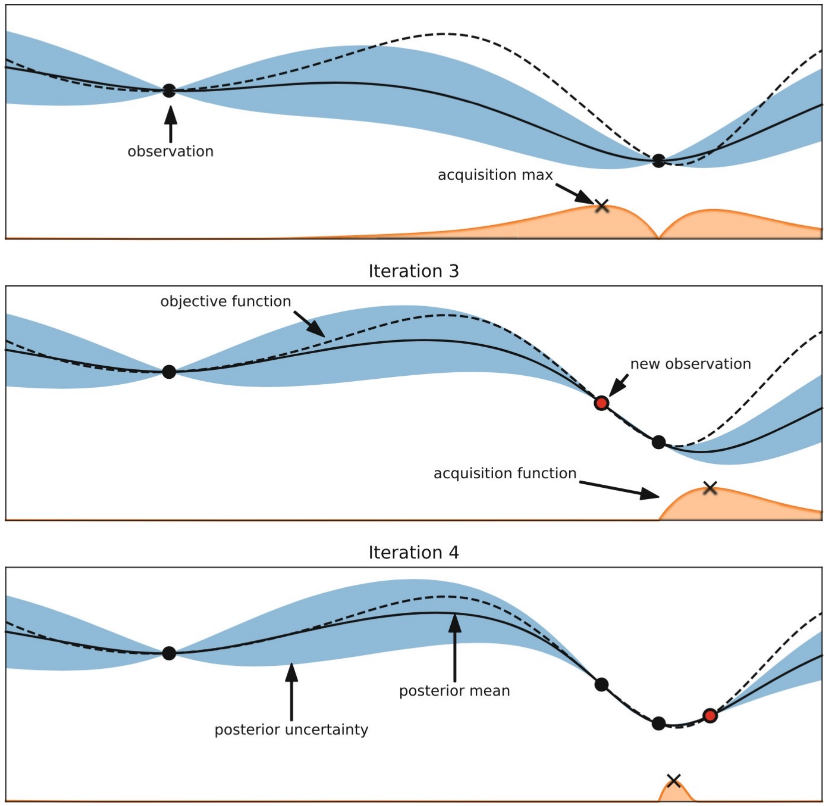
Second wave (2012-2017)

- Sequential Model Based Optimization
- Formulation of the CASH (Combined Algorithm Selection and Hyperparameter optimization) problem (quite similar to FMS)
- Multi objective approaches to FMS
- Meta learning + Optimization

Bayesian optimization

- A global optimization procedure, based on two functions:
  - Surrogate,
    - Gaussian processes, random forest, etc
  - Acquisition functions

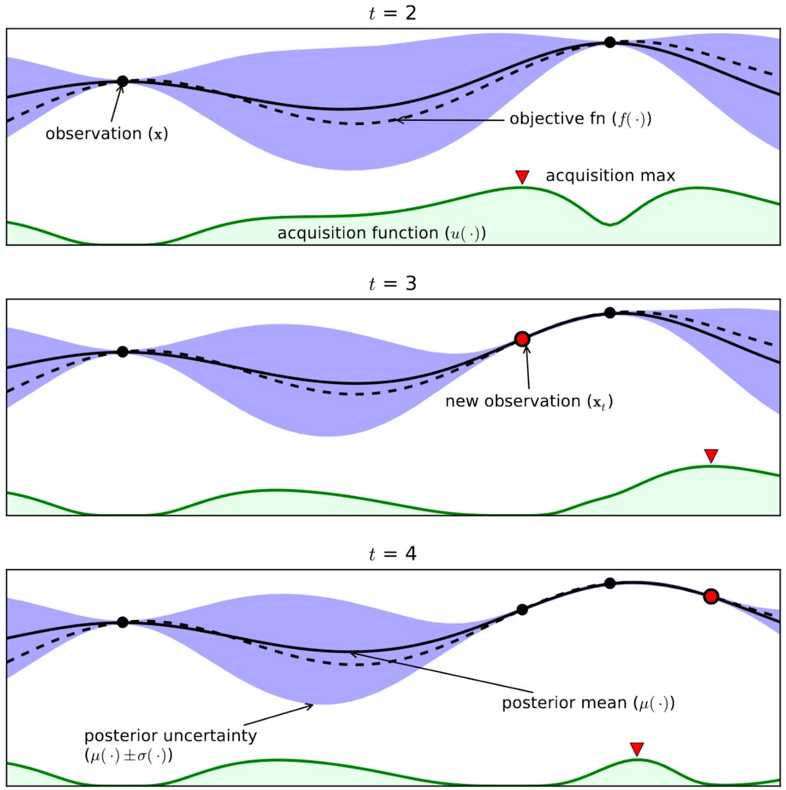
$$\mathbb{E}[\mathbb{I}(\boldsymbol{\lambda})] = \mathbb{E}[\max(f_{min} - y, 0)]$$



Feurer M., Hutter F. (2019) **Hyperparameter Optimization**. In: Hutter F., Kotthoff L., Vanschoren J. (eds) Automated Machine Learning. The Springer Series on Challenges in Machine Learning. Springer, Cham

AutoWeka

- Introduced the definition of CASH, and approached the problem with Bayesian Optimization / Sequential Model-Based Optimization (SMBO)
  - BO/SMBO: sequential design strategy for global optimization of black-box functions that doesn't require derivatives.

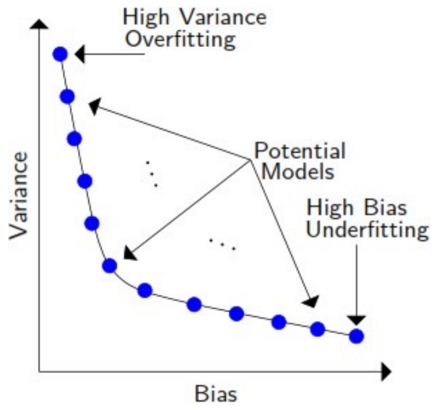


https://www.cs.ubc.ca/labs/beta/Projects/autoweka/

C. Thornton, F. Hutter, H. Hoos, and K. Leyton-Brown. **Auto-WEKA: combined selection and hyperparameter optimization of classification algorithms.** In Proc. KDD'13, pages 847–855, 2013.

Multi objective FMS

- Selecting models that optimize more than a single criterion combinations include:
  - Bias-variance
  - Performance-time
  - Performance-complexity

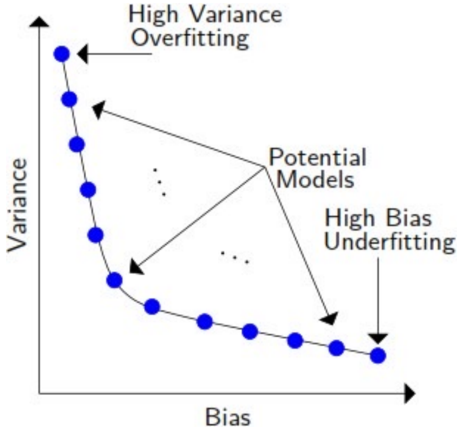


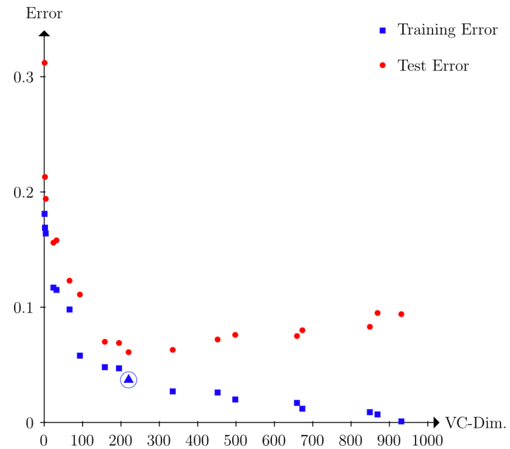
$$\text{minimize} \quad \mathbf{f}(\mathbf{x}) = [f_1(\mathbf{x}), \ldots, f_l(\mathbf{x})]^T$$
$$\text{subject to} \quad \mathbf{x} \in \mathcal{X}$$

Alejandro Rosales-Pérez, Hugo Jair Escalante, Jesús A. González, Carlos A. Reyes, Carlos A. Coello. **Bias-variance multi-objective optimization for svms parameter selection .** IbPRIA 2013: 6th Iberian Conference on Pattern Recognition and Image Analysis, Madeira, Portugal. June 5-7, 2013

Multi objective FMS

- Selecting models that optimize more than a single criterion





Surrogate models were used here!

Alejandro Rosales-Pérez, Jesus A. Gonzalez, Carlos A. Coello Coello, Hugo Jair Escalante, Carlos A. Reyes García: **Multi-objective model type selection.** Neurocomputing 146: 83-94 (2014)

Alejandro Rosales-Pérez a,n, Jesus A. Gonzalez a, Carlos A. Coello Coello b, Hugo Jair Escalante a, Carlos A. Reyes-Garcia. **Surrogate-assisted multi-objective model selection for support vector machines.** Neurocomputing 150 (2015) 163–172, 2015

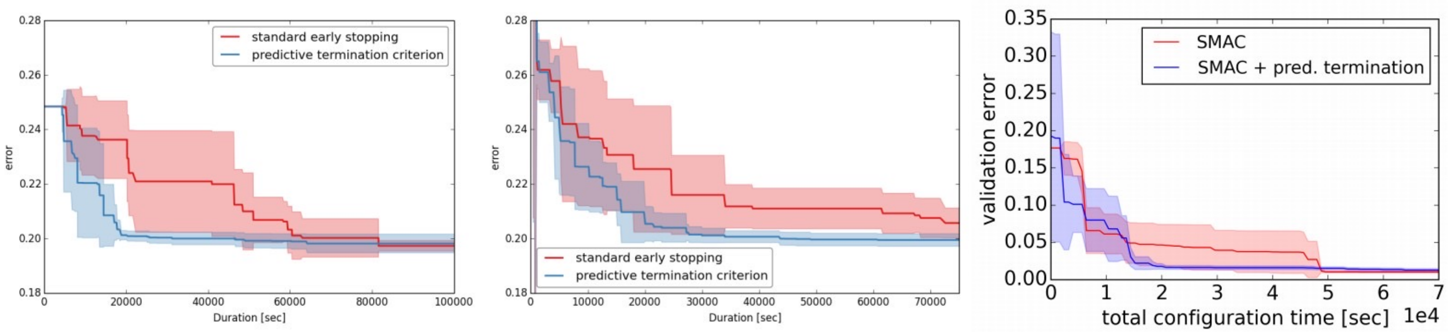
Multi fidelity approaches

- For AutoML solutions based on black box optimization, the evaluation of a candidate solution (model) is computationally expensive, and usually the more models are evaluated the better the performance of the AutoML methodology

- Methods aiming to make budget-constraint approximations of the real performance have been proposed

Multi fidelity approaches

- Learning curve-based performance:
  - Build and try to predict a learning curve (e.g., model performance vs. dataset size or model complexity)
  - Discard the model whenever the predictive curve is not promising



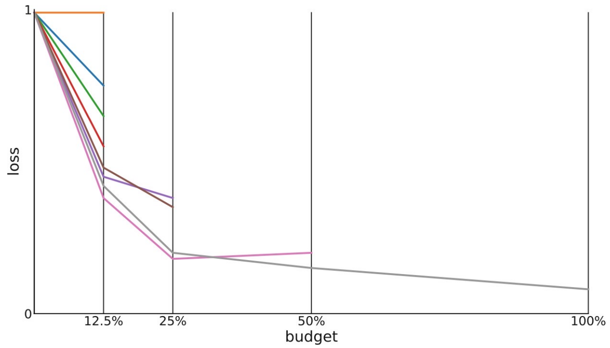
(a) SMAC on k-means CIFAR-10 (b) TPE on k-means CIFAR-10 (c) SMAC on MNIST

Tobias Domhan, Jost Tobias Springenberg, Frank Hutter **Speeding up Automatic Hyperparameter Optimization of Deep Neural Networks by Extrapolation of Learning Curves.** Proc. IJCAI 2015

Multi fidelity approaches

- **Successive halving (SH):** Given a budget (time, other resources), evaluate all algorithms for that budget; then remove the half worst models from consideration; duplicate the budget and repeat until a single model remains.

- **Hyperband:** Generate random configurations of different budgets. Then using SH as subroutine



Algorithm 1: HYPERBAND algorithm for hyperparameter optimization.

input : R, η (default $\eta = 3$)
initialization : $s_{\max} = \lfloor \log_\eta(R) \rfloor$, $B = (s_{\max} + 1)R$
for $s \in \{s_{\max}, s_{\max} - 1, \ldots, 0\}$ **do**
  $n = \lceil \frac{B}{R} \frac{\eta^s}{(s+1)} \rceil$, $r = R\eta^{-s}$
  // begin SUCCESSIVEHALVING with (n, r) inner loop
  T =get_hyperparameter_configuration(n)
  for $i \in \{0, \ldots, s\}$ **do**
    $n_i = \lfloor n\eta^{-i} \rfloor$
    $r_i = r\eta^i$
    $L = \{$run_then_return_val_loss$(t, r_i) : t \in T\}$
    T =top_k$(T, L, \lfloor n_i/\eta \rfloor)$
  end
end
return *Configuration with the smallest intermediate loss seen so far.*

K. Jamieson and Talwalkar. A. Non-stochastic Best Arm Identification and Hyperparameter Optimization. 2016
L. Li et al. Hyperband: **A Novel Bandit-Based Approach to Hyperparameter Optimization.** JMLR 2018

Meta-learning: learning to learn

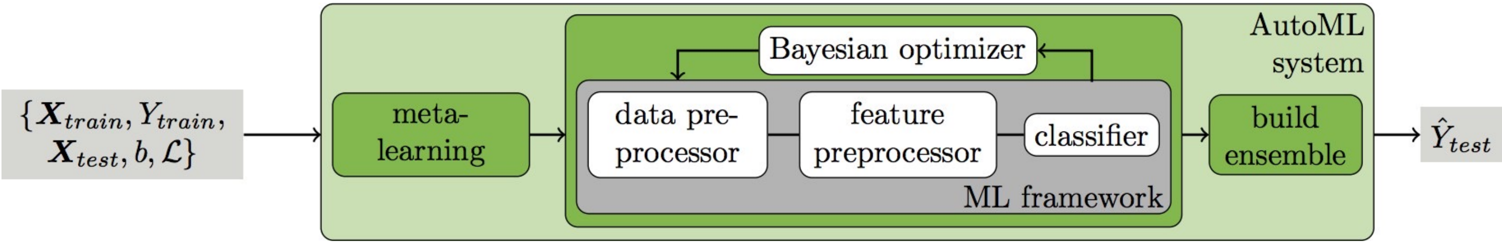
- Evaluates and compares the application of learning algorithms on (many) previous tasks/domains to suggest learning algorithms (combinations, rankings) for new tasks
- Focuses on the relation between tasks/domains and learning algorithms
- Accumulating experience on the performance of multiple applications of learning methods

Brazdil P., Giraud-Carrier C., Soares C., Vilalta R. **Metalearning: Applications to Data Mining.** Springer Verlag. ISBN: 978-3-540-73262-4, 2008.

Brazdil P., Vilalta R, Giraud-Carrier C., Soares C.. **Metalearning.** Encyclopedia of Machine learning. Springer, 2010.

AutoSKlearn

- Meta-learning + SMBO
- Trained a meta-learner on OpenML, uses it to warmstart the optimization process
- Ensembles



Matthias Feurer, Aaron Klein, Katharina Eggensperger, Jost Tobias Springenberg, Manuel Blum, and Frank Hutter. Auto-sklearn: Efficient and Robust Automated Machine Learning, pages 113–134. Springer International Publishing, Cham, 2019.

Main developments from the second wave

- Surrogate models
- Multi objective AutoML
- Bayesian optimization is now a “standard”
- Multi fidelity approaches
- Resurgence of meta-learning
- …

Three waves of AutoML

2006-2011

Full model selection, first surrogates, ensembles of solutions

2012 - 2017

SMBO, AutoSKlearn, AutoWEKA, TPOT

2018 - ?

Neural Architecture Search, RL

Renascence of Meta Learning

Few shot learning, etc.

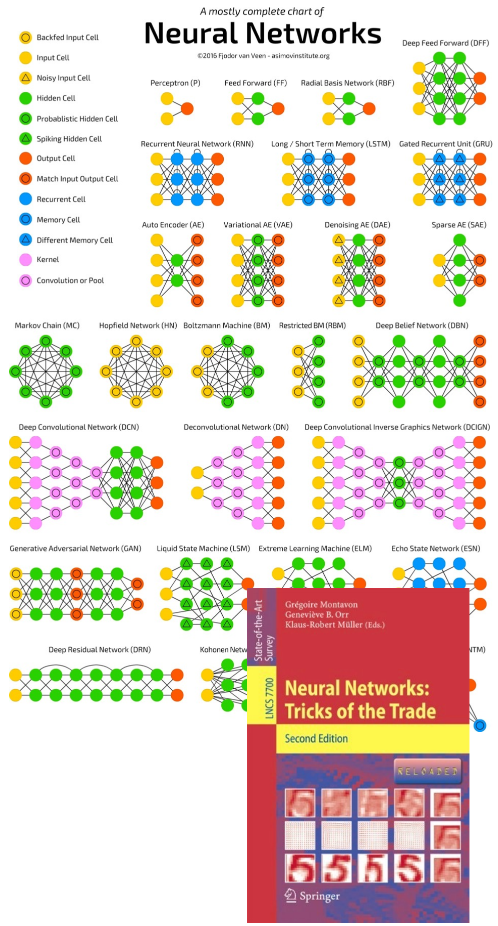
Third wave (2018 - and on ...)

- Neural architecture search
- Reinforcement learning based AutoML
- Reinassence of evolutionary algorithms for AutoML
- AutoML from raw data!

Neural architecture search

- It is well known that the success of deep learning (DL) solutions relies heavily on the design choices and hyperparameters

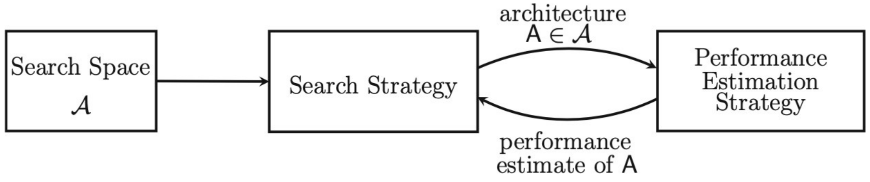
- DL is everywhere, hence, AutoML methods targeting DL can have a huge impact



Neural architecture search

- NAS is the process of automating architecture design in the context of DL models (share same complications as AutoML in supervised learning)

- Components of a NAS method:
  - Search space
  - Search strategy
  - Performance estimation strategy



T. Elsken et al.. (2019) **Neural architecture search**. In: Hutter F., Kotthoff L., Vanschoren J. (eds) Automated Machine Learning. The Springer Series on Challenges in Machine Learning. Springer, Cham

Neural architecture search

- Search strategy:
  - Evolutionary algorithms
  - Bayesian optimization
  - Reinforcement learning
  - Gradient based

- Performance estimation:
  - Multi fidelity approaches
  - One shot architecture search

Latest developments

- Neural architecture search is driving new ways for efficient AutoML
- RL-based AutoML
- Meta learning resurgence
- Complex search spaces that require of new optimization methodologies

Final remarks

AutoML / Full model selection

- Pros
  - The job of the data analyst is considerably reduced
  - Neither knowledge on the application domain nor on machine learning is required
  - Different methods for preprocessing, feature selection and classification are considered
  - It can be used in any classification problem

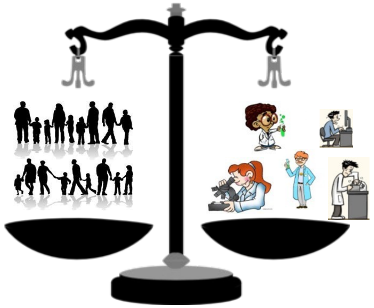
- Cons
  - It is real function + combinatory optimization problem
  - Computationally expensive
  - Risk of overfitting

NLF theorem!

… and full models for all

- **Short-term goal:** to provide data analysts with tool that allows them to build effective classification systems without much effort

- **Long-term goal:** An APP that allows anyone to build a classification model from their data (photographs, smart phone data, tweets, etc.)

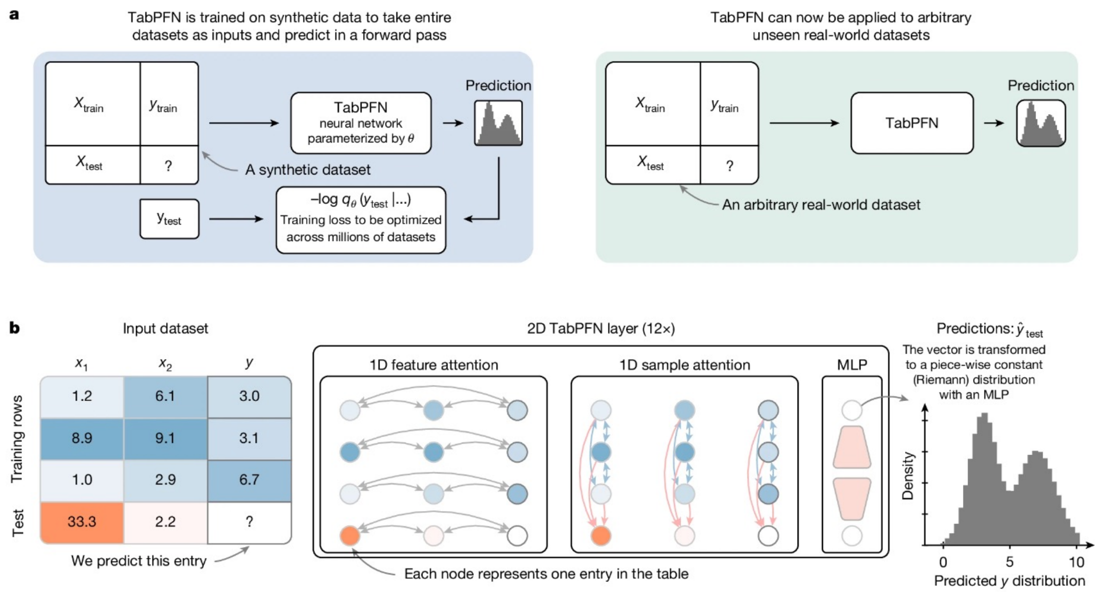


Challenges and research opportunities

- Explainable AutoML models.
- AutoML in feature engineering.
- AutoML for non tabular data.
- Large scale AutoML.
- Transfer learning in AutoML.
- Benchmarking and reproducibility in AutoML.
- Interactive AutoML methods.

Foundational Models of Tabular Data Learning

- **TabPFN (Tabular Prior-Data Fitted Network)** is a transformer-based foundation model designed for fast and accurate classification of tabular datasets.
- Key Features:
  - Ultra-fast predictions in under a second without hyperparameter tuning.
  - Supports both numerical and categorical data.
  - Works effectively for datasets with up to 1,000 samples, 100 features, and 10 classes.
  - Scikit-learn compatible interface for easy integration.



https://arxiv.org/abs/2207.01848

Foundational Models of Tabular Data Learning

- **Training Process:**
  - TabPFN is trained on millions of synthetic datasets generated from a causal prior.
  - This prior reflects realistic patterns found in real-world data.
- **Prediction:**
  - At inference, the model approximates Bayesian posterior predictions.
  - It outputs class probabilities without needing iterative training.
- **Architecture:**
  - Based on a transformer model adapted for tabular data.

Foundational Models of Tabular Data Learning

- **Performance:**
  - Matches or surpasses gradient boosting and AutoML systems on small datasets.
  - Robust even with missing values and feature noise.
- **Advantages:**
  - No hyperparameter tuning required.
  - Low computational cost.
  - Easy deployment via Python packages or APIs.

Use Cases:

Ideal for quick prototyping, me
structured data tasks.

```python
import numpy as np
import pandas as pd
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, classification_report
from tabpfn import TabPFNClassifier

# Load the Iris dataset
iris = load_iris()
X = iris.data
y = iris.target

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# Initialize the TabPFN classifier
model = TabPFNClassifier(device="cpu")  # Change to "cuda" if using a GPU

# Train the model (TabPFN is designed for quick inference without traditional training)
model.fit(X_train, y_train)

# Make predictions on the test set
y_pred = model.predict(X_test)

# Evaluate the performance
accuracy = accuracy_score(y_test, y_pred)
report = classification_report(y_test, y_pred, target_names=iris.target_names)

# Display results
print(f"Accuracy: {accuracy:.4f}\n")
```

Foundational Models of Tabular Data Learning

```python
import numpy as np
import pandas as pd
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, classification_report
from tabpfn import TabPFNClassifier

# Load the Iris dataset
iris = load_iris()
X = iris.data
y = iris.target

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# Initialize the TabPFN classifier
model = TabPFNClassifier(device="cpu")  # Change to "cuda" if using a GPU

# Train the model (TabPFN is designed for quick inference without traditional training)
model.fit(X_train, y_train)

# Make predictions on the test set
y_pred = model.predict(X_test)

# Evaluate the performance
accuracy = accuracy_score(y_test, y_pred)
report = classification_report(y_test, y_pred, target_names=iris.target_names)

# Display results
print(f"Accuracy: {accuracy:.4f}\n")
```