HyPSTER

Hyper**P**arameter Optimization On **STER**oids

Plan for today

Motivation

Machine Learning Pipeline Optimization (MLPO)

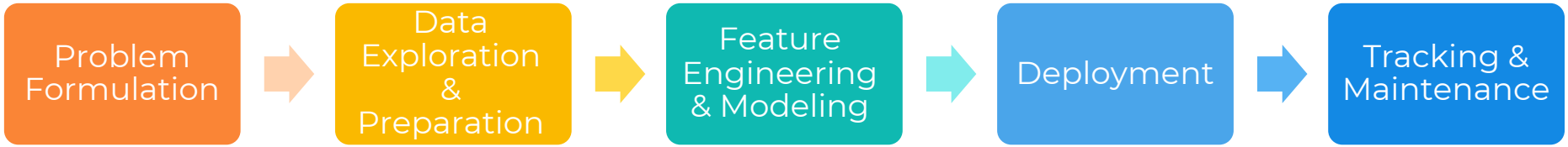
HyperParameter Optimization (HPO) Frameworks & AutoML

HyPSTER – Architechture, Features, Code

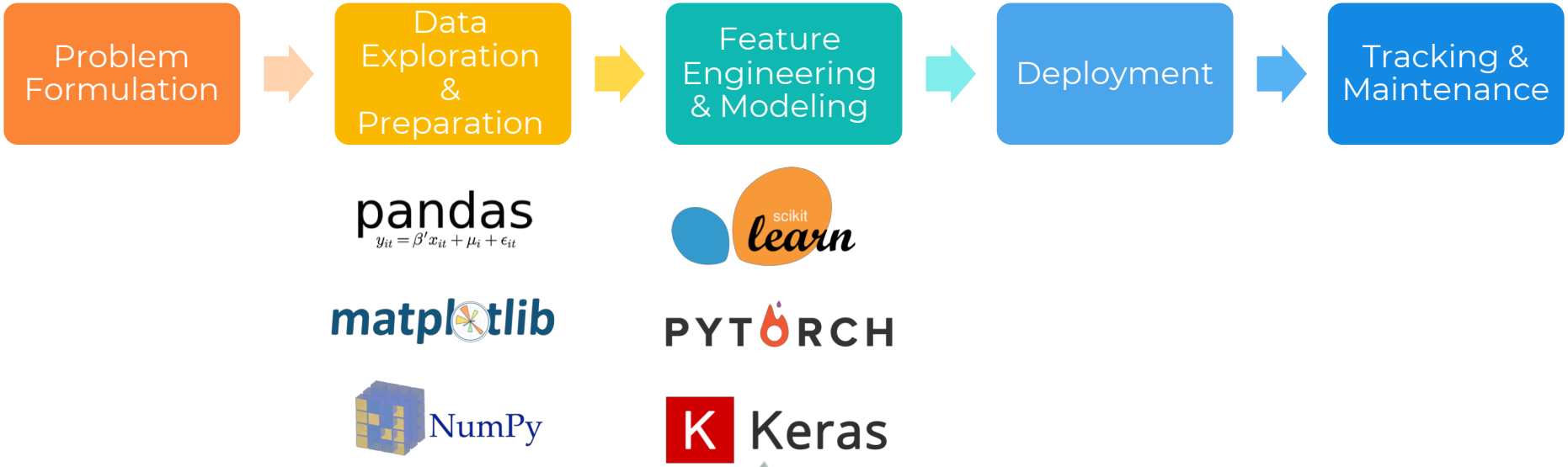
Future Plans

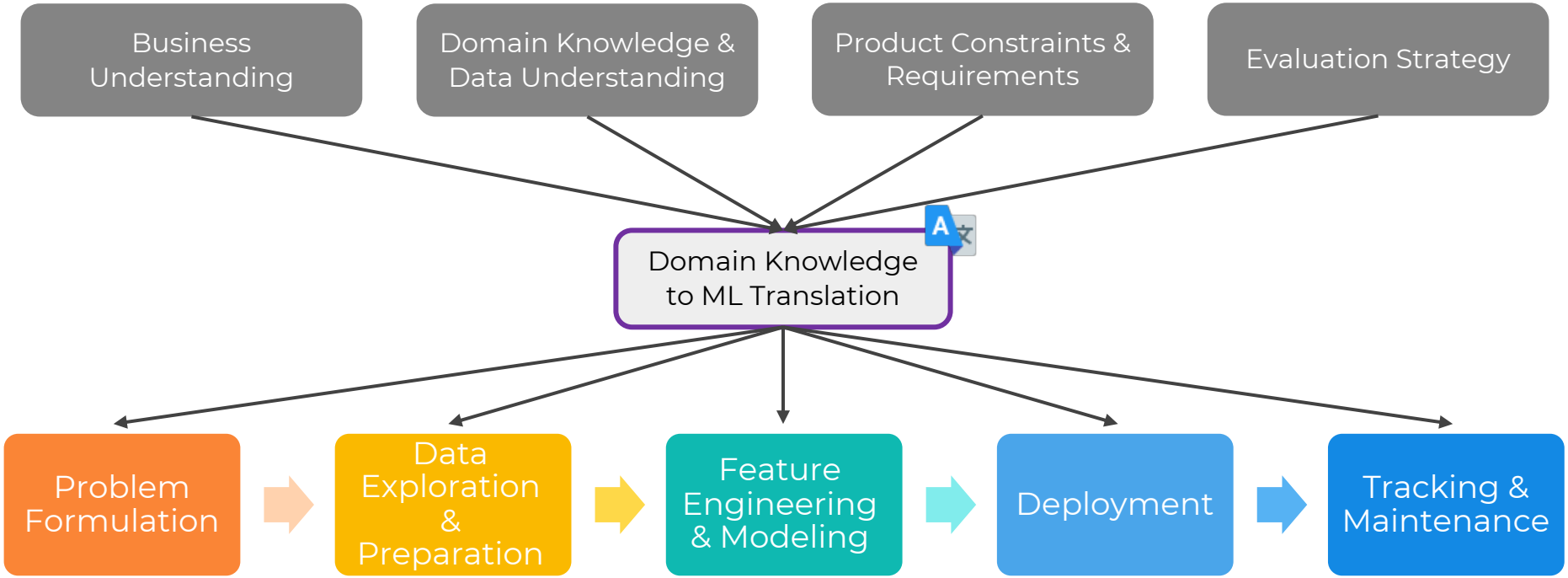
Why am I here?

Realization: It's **really** hard to make successful DS Projects



Something is missing

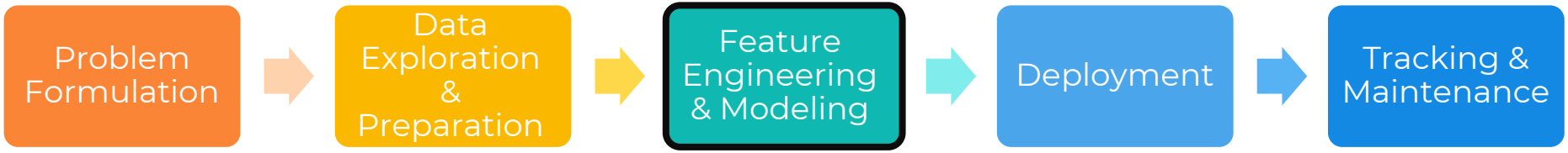



Business Understanding
Domain Knowledge & Data Understanding
Product Constraints & Requirements
Evaluation Strategy
Domain Knowledge to ML Translation
Problem Formulation
Data Exploration & Preparation
Feature Engineering & Modeling
Deployment
Tracking & Maintenance

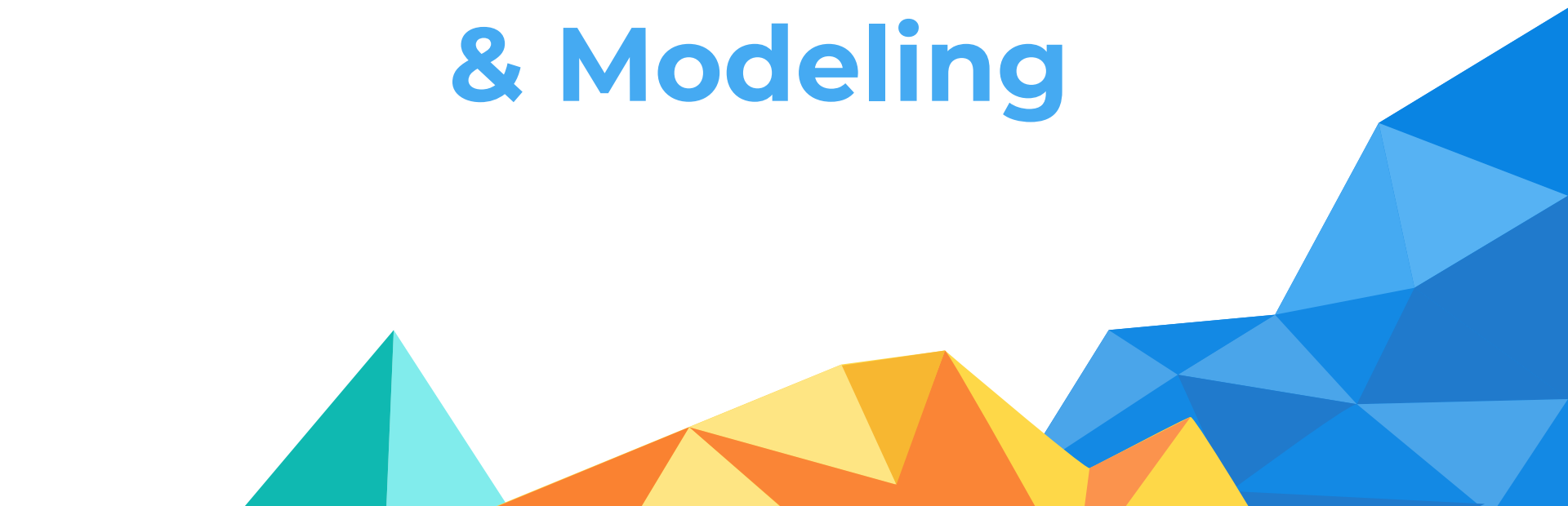
How do I get better at it?

- Not enough resources
- Time & Focus, Trial & Error
- We need to:
    A. Build New Courses & Resources
    B. Automate Everything We Can

One step at a time



Feature Engineering & Modeling



Feature Engineering & Modeling as MLPO

- **M**achine **L**earning **P**ipeline **O**ptimization
- 2 Main Components:
  A. Transformers
  B. Estimators

Null Value Imputation
Categorical Encoding
Dimensionality Reduction
Feature Interactions

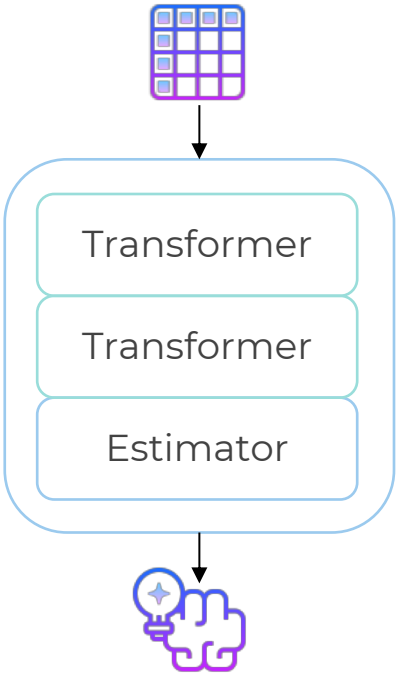
Feature Engineering & Modeling as MLPO

- **M**achine **L**earning **P**ipeline **O**ptimization
- 2 Main Components:
  A. Transformers
  B. Estimators

Random Forests
Linear Regression
XGBoost
Neural Networks

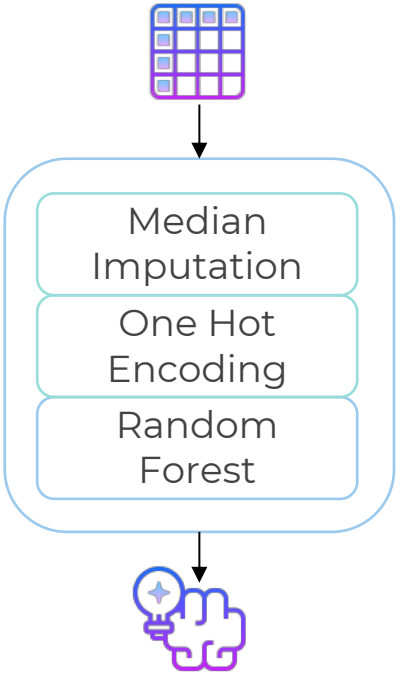
Feature Engineering & Modeling as MLPO

- **M**achine **L**earning **P**ipeline **O**ptimization
- 2 Main Components:
  A. Transformers
  B. Estimators
- Pipeline



Feature Engineering & Modeling as MLPO

- **M**achine **L**earning **P**ipeline **O**ptimization
- 2 Main Components:
  A. Transformers
  B. Estimators
- Pipeline



Why should we care?

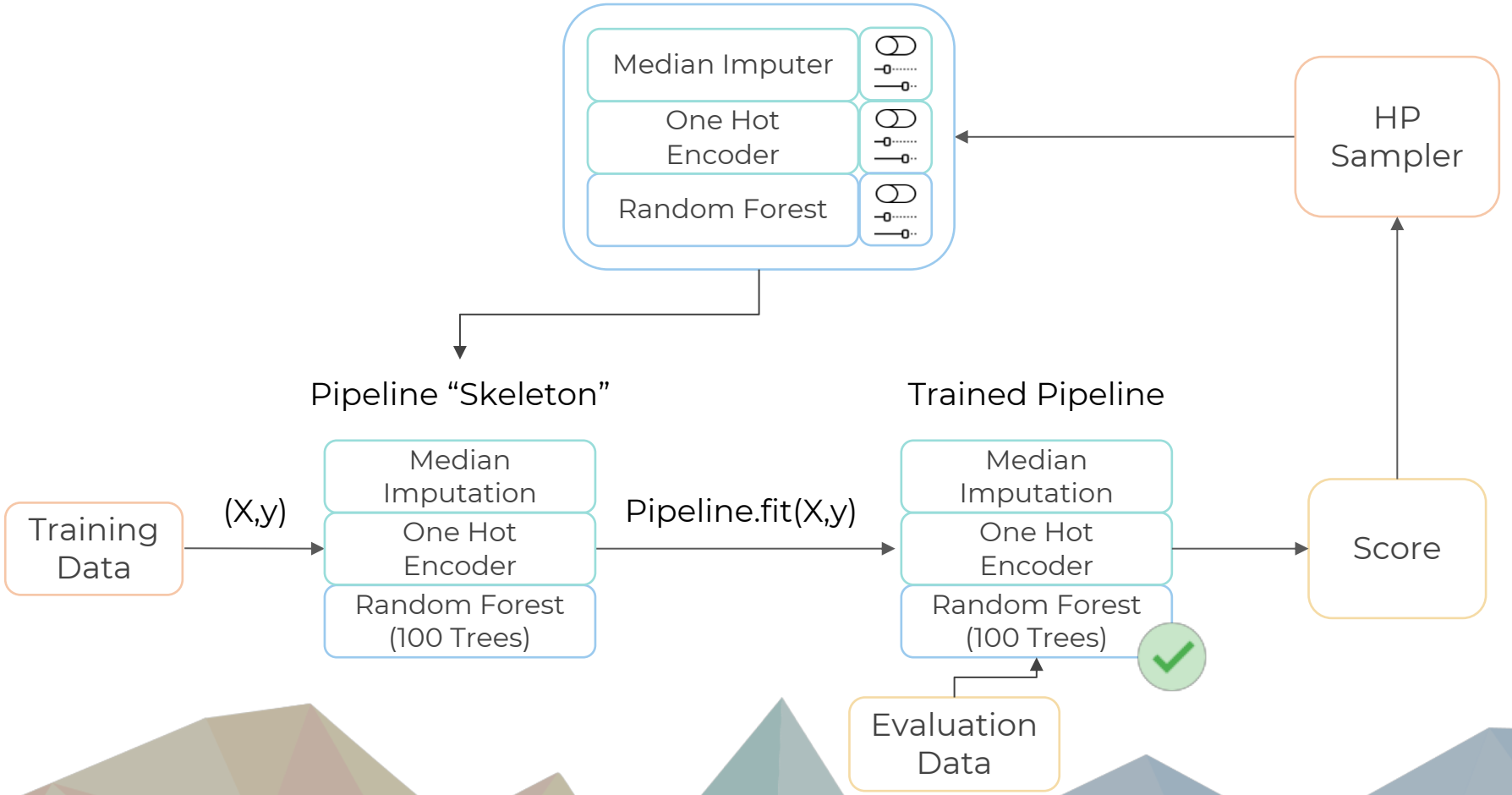
- Choosing the right ML Pipeline can significantly affect:
  - Accuracy
  - Speed
  - Hardware & Software compatibility
  - Explainability

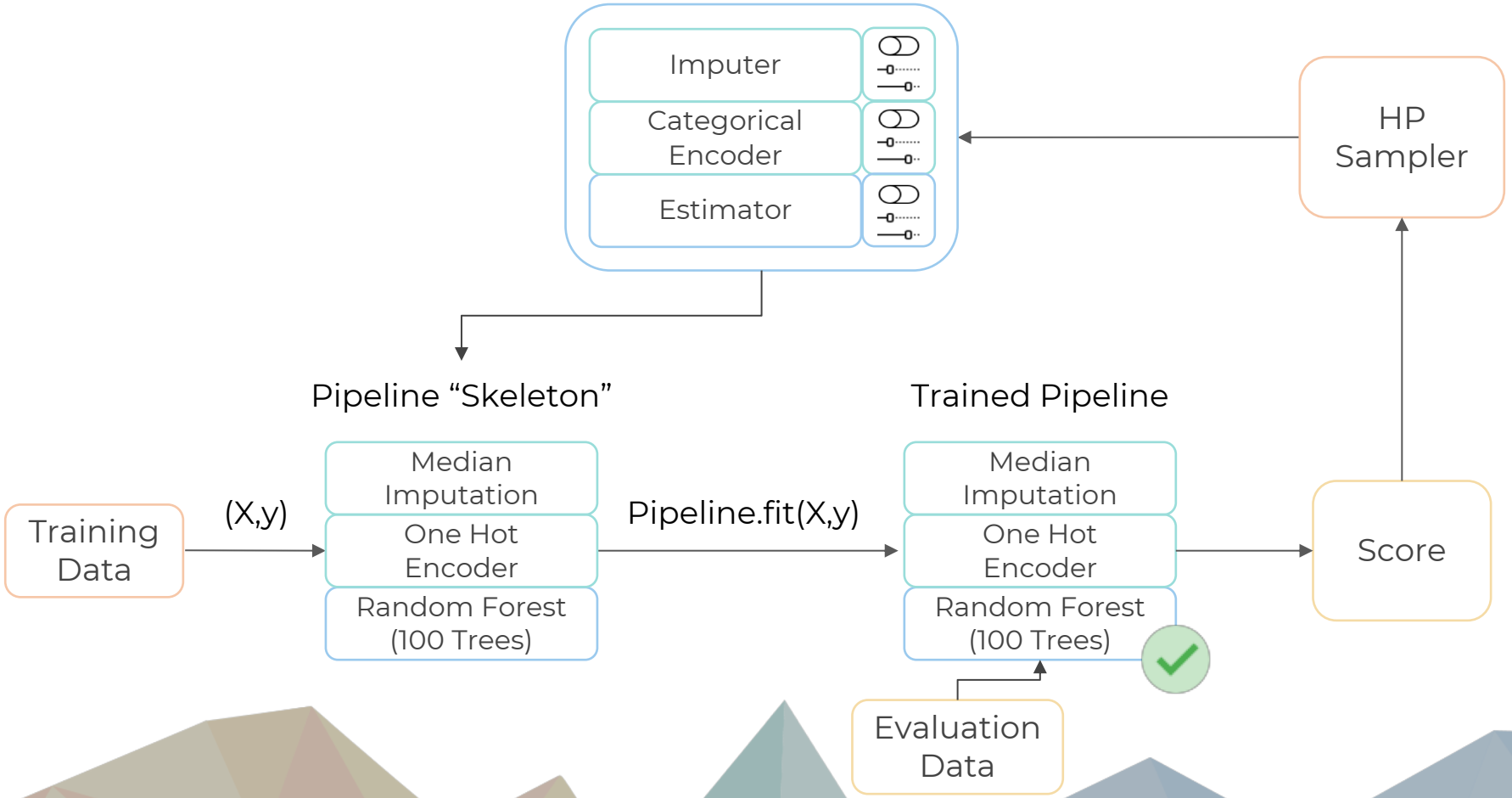
Hyper Parameters

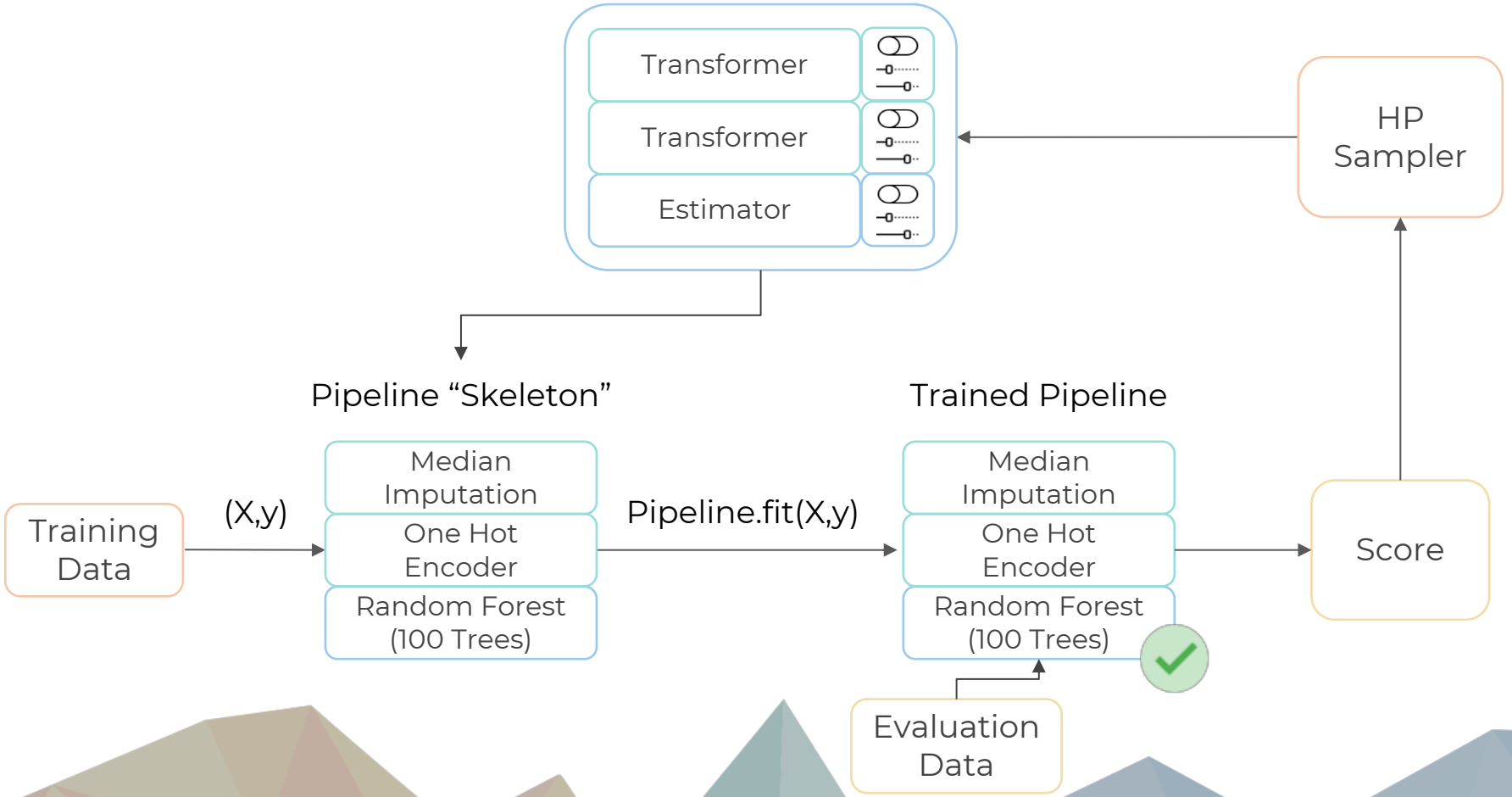
- Each component can be instantiated by HyperParameters
- Numeric or categorical
  - **PCA** – `n_components: [1,n], SVD_solver: {full,arpack}`
  - **Neural Networks** – `drop_rate: [0,1], optimizer {SGD,Adam}`

Parameters Vs. HyperParameters

- **Each Component <u>can</u> learn its optimal internal Parameters** from the training data. For example:
  A. Splits in a decision tree
  B. Weights in a Neural Network
- **Components <u>cannot</u> learn their HyperParameters** from the data directly
- How do we find optimal HyperParameters?


Median Imputer
One Hot Encoder
Random Forest
HP Sampler
Pipeline “Skeleton”
Trained Pipeline
Training Data
(X,y)
Median Imputation
One Hot Encoder
Random Forest (100 Trees)
Pipeline.fit(X,y)
Median Imputation
One Hot Encoder
Random Forest (100 Trees)
Score
Evaluation Data

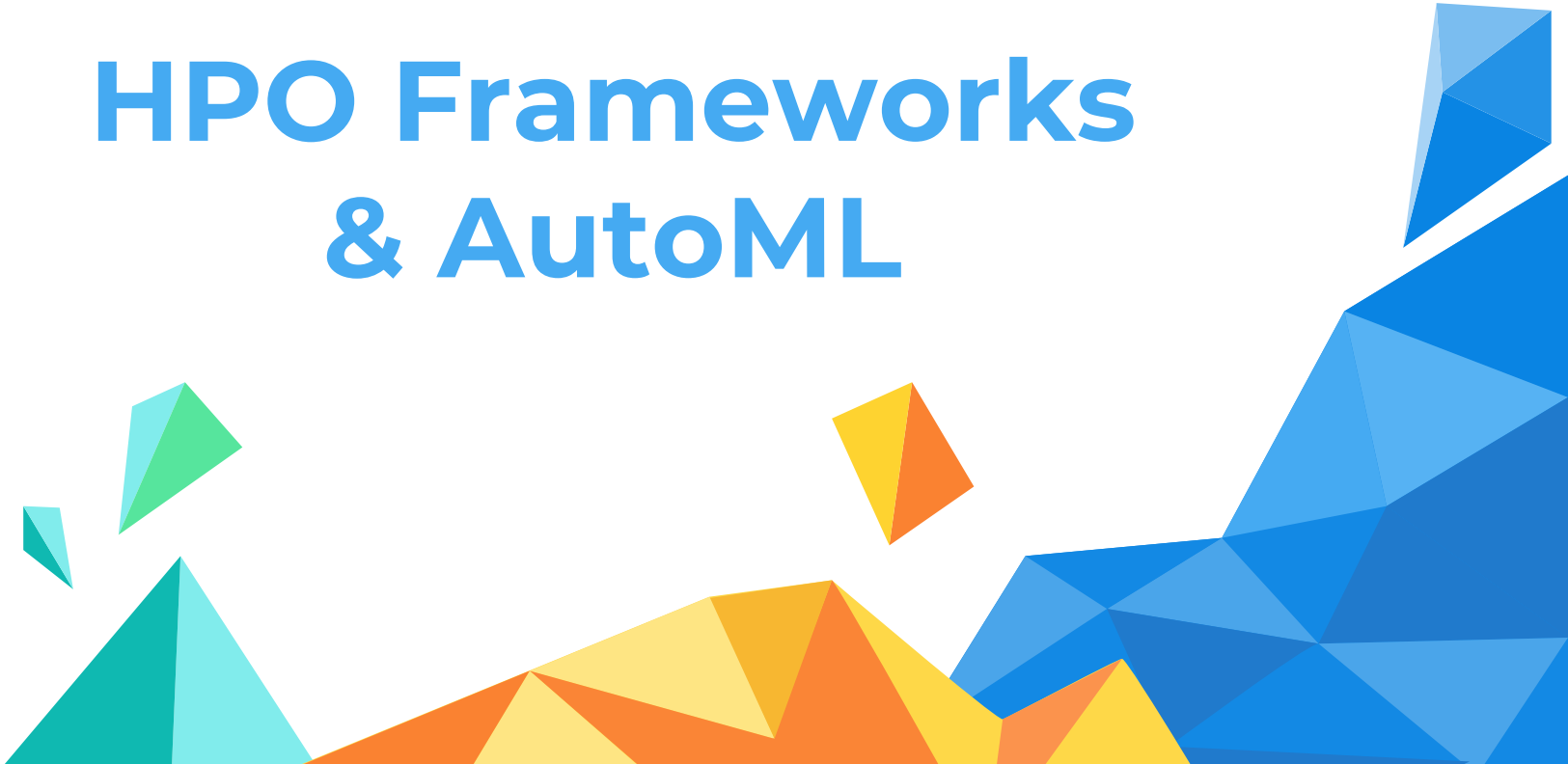

Imputer
Categorical Encoder
Estimator
HP Sampler
Pipeline “Skeleton”
Trained Pipeline
Training Data
(X,y)
Median Imputation
One Hot Encoder
Random Forest (100 Trees)
Pipeline.fit(X,y)
Median Imputation
One Hot Encoder
Random Forest (100 Trees)
Score
Evaluation Data


Transformer
Transformer
Estimator
HP Sampler
Pipeline “Skeleton”
Trained Pipeline
Median Imputation
One Hot Encoder
Random Forest (100 Trees)
Training Data
(X,y)
Pipeline.fit(X,y)
Median Imputation
One Hot Encoder
Random Forest (100 Trees)
Score
Evaluation Data

Our Goal

- To choose the **right components** in the **right order** with the **right HPs** that are optimal w.r.t. our evaluation metric

HPO Frameworks & AutoML



HPO Frameworks

- Optuna, HyperOpt, GPyOpt, Scikit-Optimize

Pros

- Very Customizable
- Advanced Algorithms

Cons

- Lots of code
- Expertise

AutoML Frameworks

- Auto-SKLearn, TPOT, H2OAutoML

Pros

- User Friendly – “One Liner”
- Accurate

Cons

- Less Customizable
- Slow
  A. Heavy Feature Engineering, Complex Pipelines
  B. Not using latest Algorithms

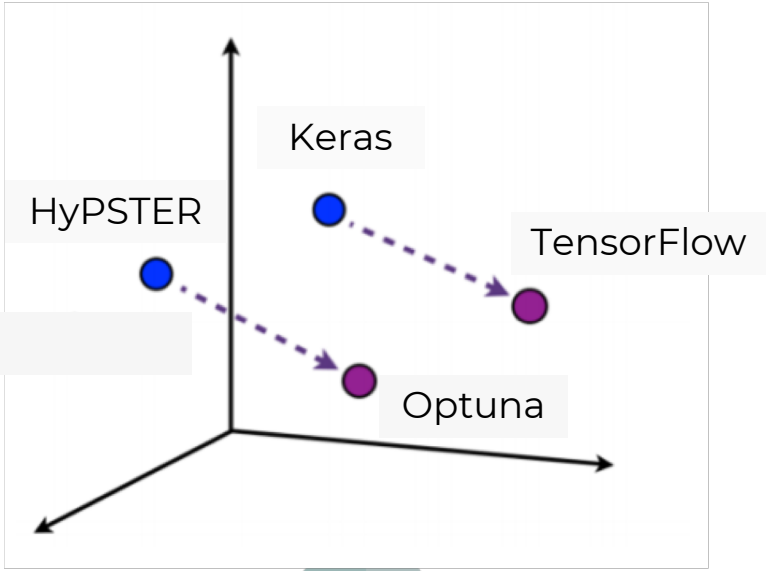
Ideally, we would want

- User Friendly – “One Liner”
- Accurate
- Fast
- Customizable

HyPSTER

HyPSTER - Overview

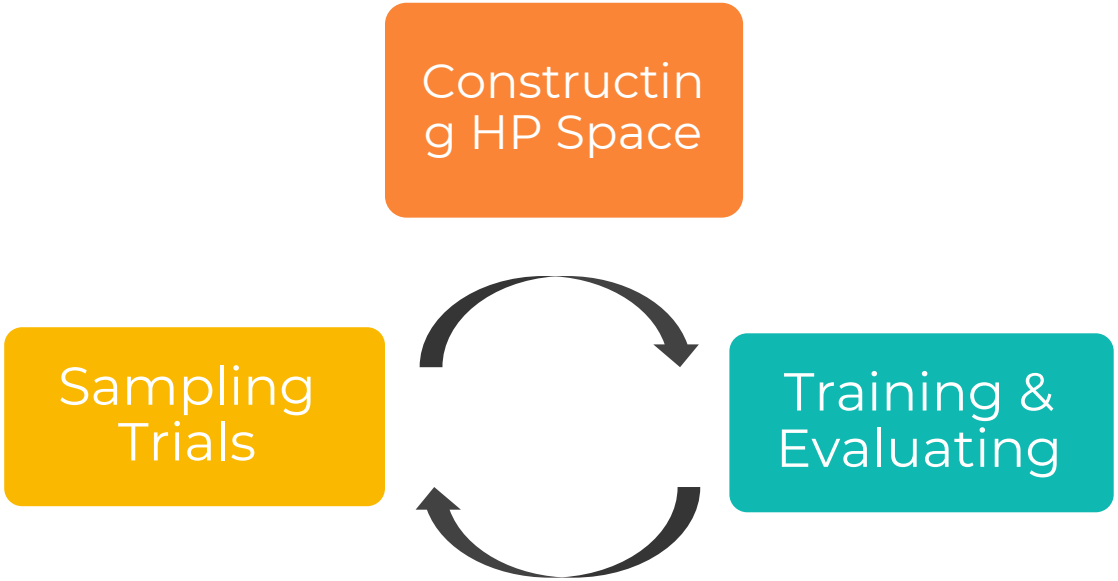
1. High-Level API for **ML Pipeline Optimization** built on top of **Optuna**



HyPSTER - Overview

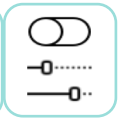
1. High-Level API for **ML Pipeline Optimization** built on top of **Optuna**
- It has 3 “modes”:
    A. **Research Mode** - Fast experimentation and feedback
    B. **Production Mode** - Great performing ML Pipelines
    C. **Competition Mode** - Gigantic multi-layered Stacked pipelines

How does it work?



HP Search Space Construction

a. Each Component has
   a. Tags
   b. “Raw” HyperParameters

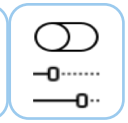
SimpleImputer 

Sparse Input: True
Interpretable: True

```
method: {“mean”, “median”, “constant”}
missing_indictor: {True, False}
```

HP Search Space Construction

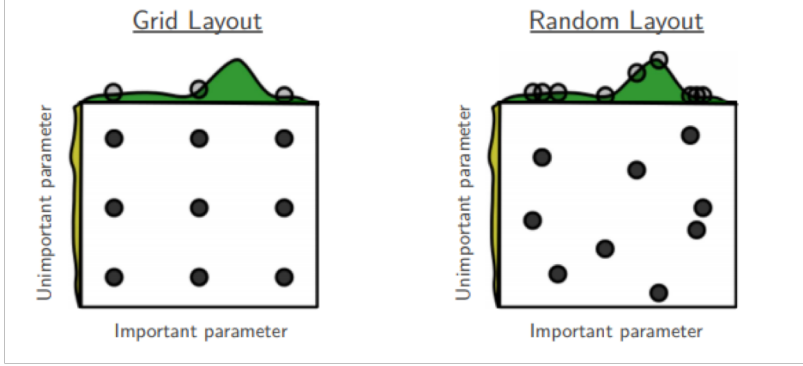
a. Each Component has
   a. Tags
   b. “Raw” HyperParameters

CatBoost 

Supports Classification: True
Handles Sparse Input: False
Handles Categorical: True

```
max_depth: int(2,20)
l2_leaf_reg: real[0.1, 5]
```

Intelligent HP Sampling



- Usually – Grid/Random Search
- Intelligent HP Sampling - Learns from previous trials
  A. TPE (Tree Parzen Estimator)
  B. CMA-ES (Covariance Matrix Adaptation Evolution Strategy)
  C. Add your own sampler or import from Scikit-Optimize

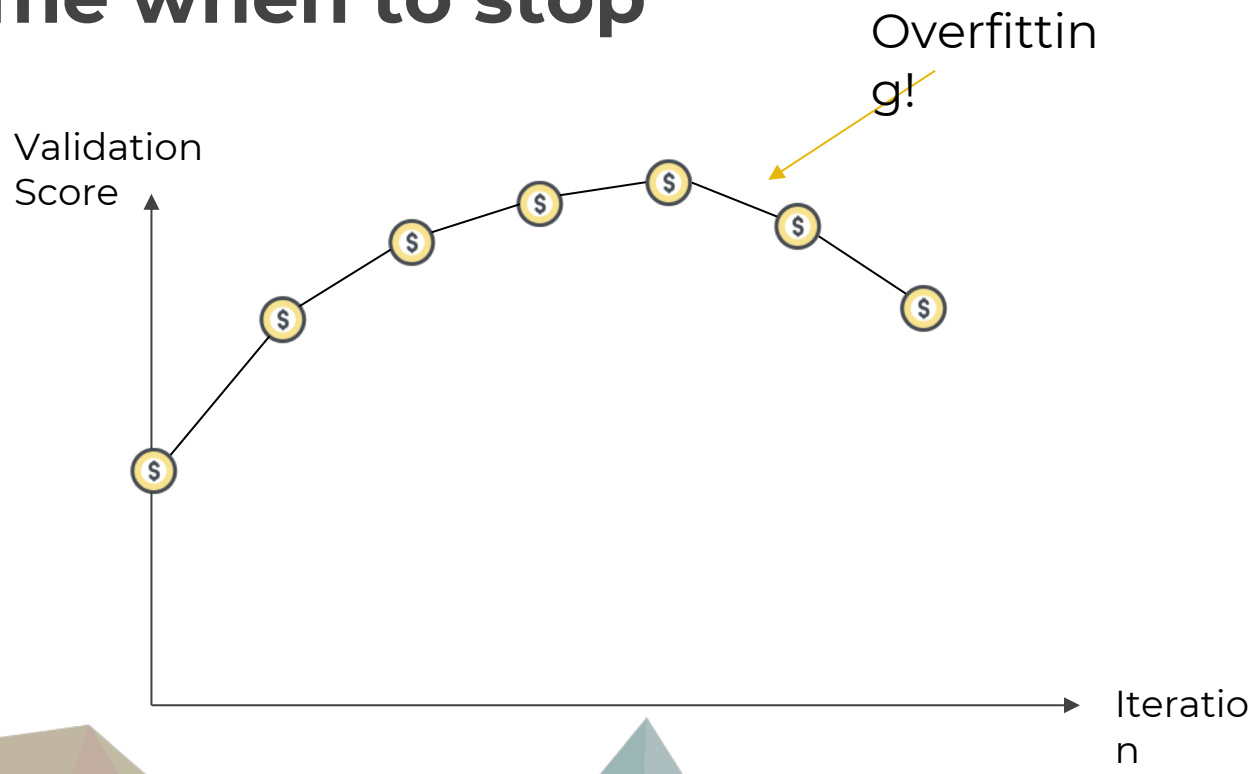
Training Phase

- We've sampled a Pipeline "Skeleton". Now we need to train it!
- **Observation**: Most modern ML models work in iterations
  A. **A Tree** in Tree-Ensembles
  B. **An Epoch** in Neural Networks
- Usually
  A. n_iterations, learning rate are "regular" HyperParameters
  B. Running n_iterations with learning_rate Checking Performance

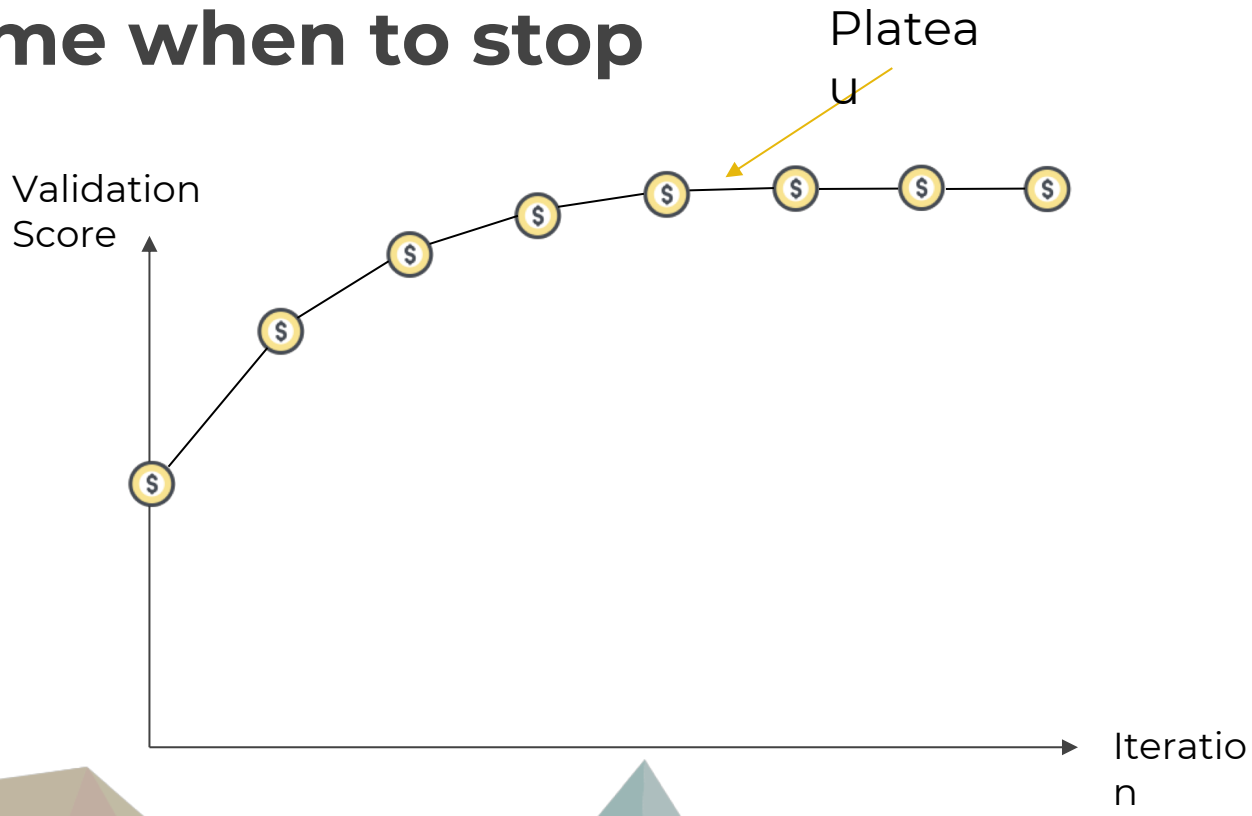
With HyPSTER – Every Iteration Counts!

- Let’s play a game
- You’ve received 1,000 coins
- You can “pay” with one coin for one training iteration
- You can stop training whenever you want

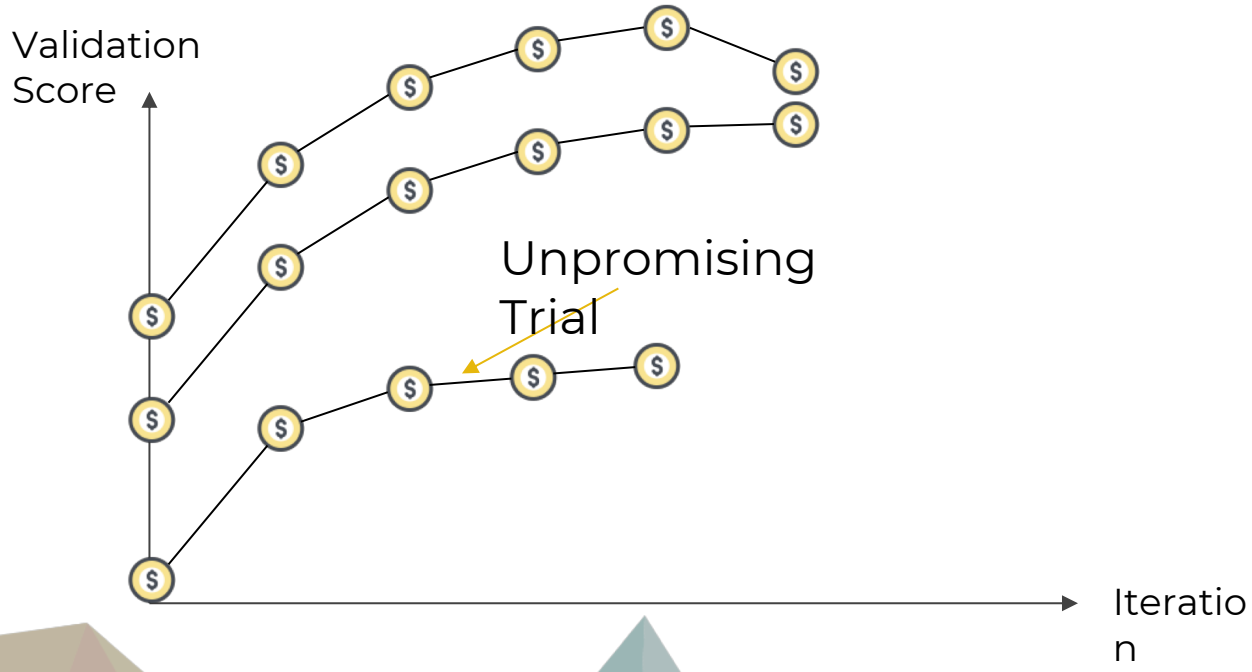
Tell me when to stop



Tell me when to stop



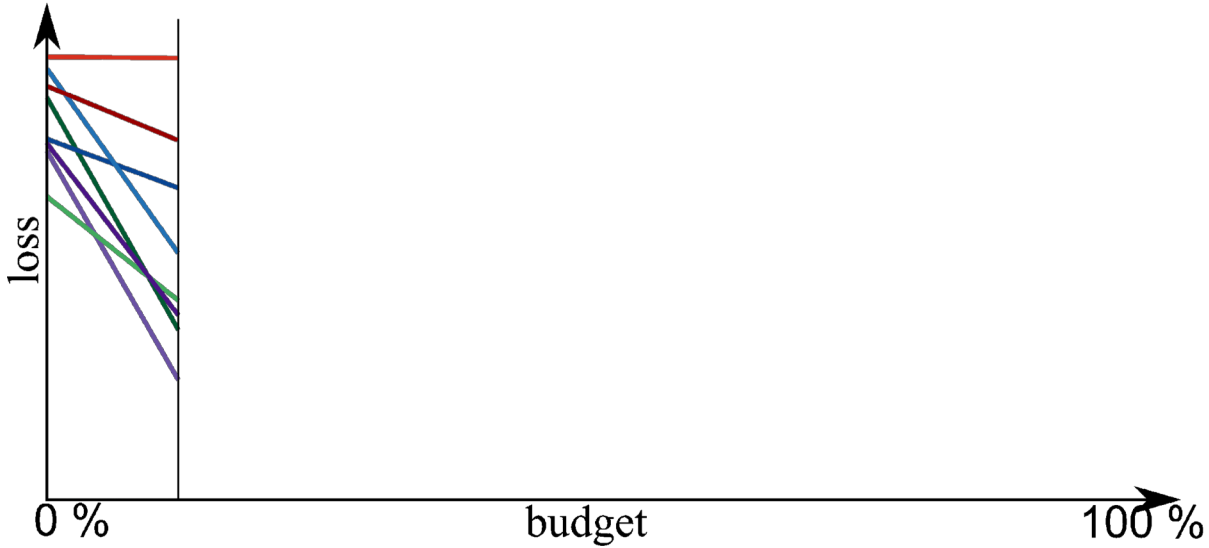
Tell me when to stop



Early Stopping & Pruning

- **Early Stopping** – Avoids Overfitting & Saves Time On Plateau
- **Pruning** – Stops training for unpromising trials (Successive Halving)
- Using Cross Validation!

Successive Halving In Action

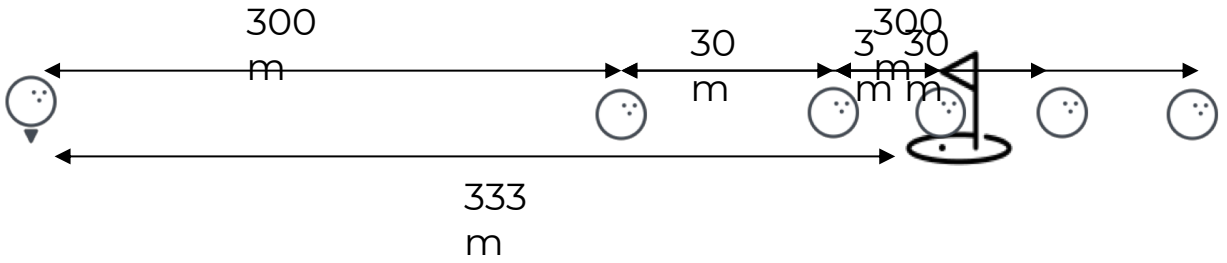


Source:
https://www.automl.org/blog_bohb/

Adaptive Learning Rate Decay

- Learning Rate = how much the model should change its parameters in the current iteration

Adaptive Learning Rate Decay



Adaptive Learning Rate: 2 + 2 + 1 = 5

Constant Learning Rate: 111

HyPSTER Features

- Supports Classification (Binary, Multiclass) & Regression
- Handles Dense & Sparse Matrices
- Outputs Sk-Learn compatible Pipelines
- Study statistics can be saved in memory or on SQL & RDB databases
- Runs in parallel (multi node CPU/GPU)

Current State of affairs

- Very early stages
- **Transformers**
  A. Imputation
  B. Categorical Encoding
  C. Scaling
- **Estimators**
  A. XGBoost
  B. SGD

Shut up and show me some code

```
xgb = XGBClassifierHypster(booster_list=["dart", "gbtree"])
sgd = SGDClassifierHypster()
estimators = [xgb, sgd]
clf = HyPSTERClassifier(estimators, scoring="roc_auc", cv=3)
clf.fit(X_train, y_train, cat_cols=cat_cols, n_trials=40)
clf.predict(X_test)
```

Future Plans

- Higher-level abstractions (Complexity, Explainability, Performance Constraints)
- Competition mode - **Stacking On Steroids**
- Live visualizations
- More Estimators & Transformers

Recap

- It's **really** hard to create successful DS Projects
- **Translating Domain Knowledge into ML Language** is a core skill
- Automation = Focusing on our added value
- HyPSTER automates large parts of **Feature Engineering & Modeling**
- Using a **High-Level Machine Learning Pipeline Optimization** API on top of Optuna
- It's fast, accurate, customizable and user friendly
- So `pip install hypster` & `.fit()` `.predict()` away

Questions?

Thanks!

gilad.rubin@gmail.com