Machine Learning in Belle II Analyses (w/ Hands-on)

T. Lam and B. Zhang

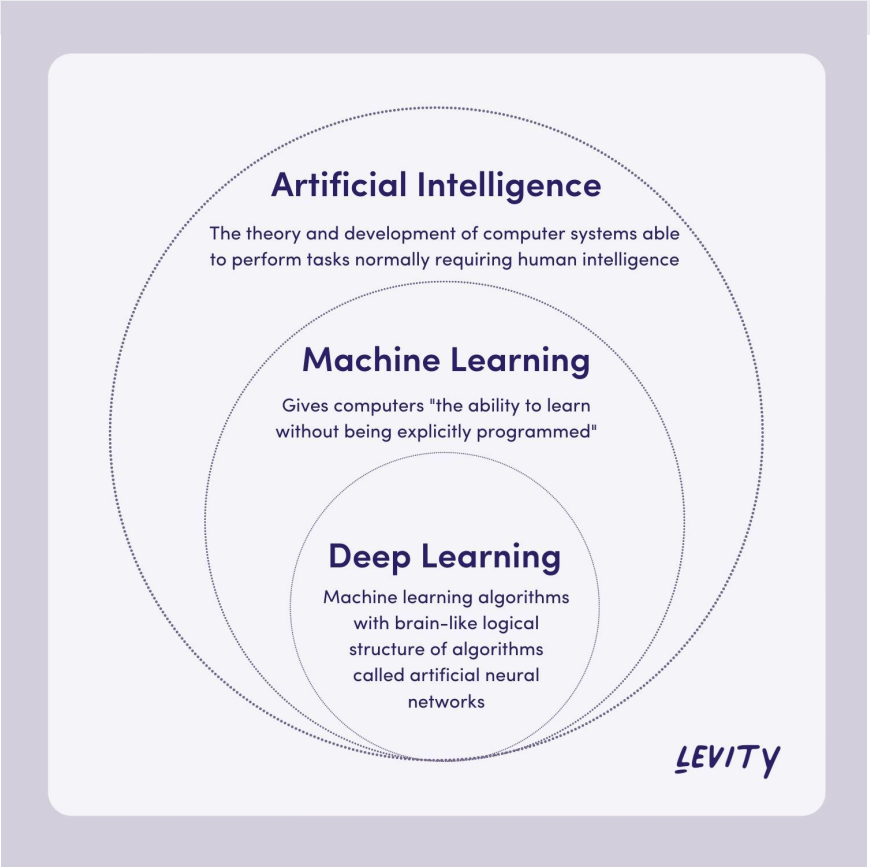
UNIVERSITY of HAWAIʻI®
MĀNOA

Outline

- Introduction of Machine Learning
  - Why do we do machine learning?
  - Examples of Usage
- How to do Machine Learning?
- Model of Interest: Decision Trees
- Hands-on exercise
  - Clustering Review
  - Kaggle time

What is Machine Learning?

- “Machine learning is a form of AI that enables a system to learn from data rather than through explicit programming.”



PC:https://levity.ai/blog/difference-machine-learning-deep-learning

Types of Machine Learning Problems (w/ ex.)

Supervised Learning

Classification

-Particle identification
-Distinguishing between signal vs. background

Regression

-Approximating Functions
-Collimator Knob Tuning

Semi-Supervised Learning

Image Generation

-PXD background overlay generation
-ECL Waveform generations

Anomaly Detection

-Computer Vision for DQM at HYDRA

Unsupervised Learning

Clustering

-Improved ECL Cluster Search
-GNN Tracking at Belle 2

Anomaly Detection

-NP Detection for Dark Higgs at Belle 2

Reinforcement Learning

Language Models

-ChatGPT
-ARICH shifter bot

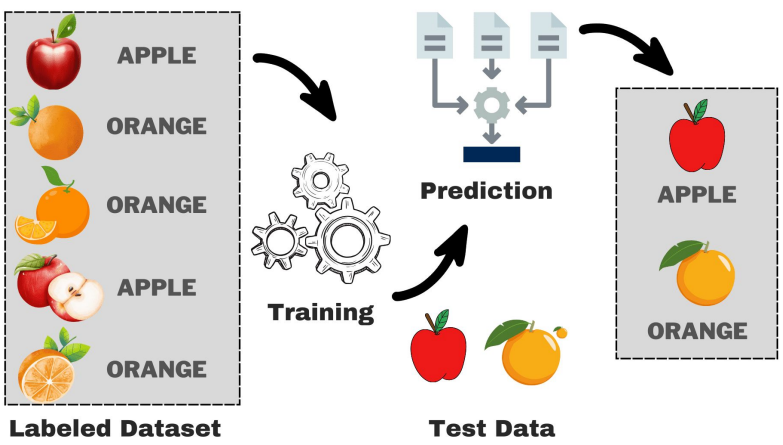
Time-Dependent

-Real-time anomaly detection during operation
-Self-Driving Cars

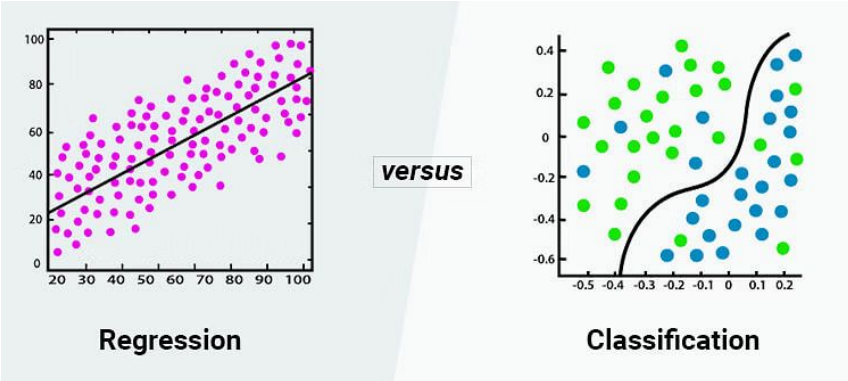
For better view of the landscape: https://iml-wg.github.io/HEPML-LivingReview/

Supervised Learning

- Given a dataset with a set of labels/indices, train your machine to get those labels right.
  - If labels are discrete, then we have classification
    - If labels are continuous, then we have regression
- If not all your dataset is labeled or you don't have labels, then you are semi-supervised or unsupervised



PC:https://www.kdnuggets.com/wp-content/uploads/mehreen_understanding_supervised_learning_theory_overview_6.png



PC:https://www.simplilearn.com/ice9/free_resources_article_thumb/Regression_vs_Classification.jpg

Examples of Recent Analyses w/ Classifiers

Observation of the decay B0 → J/ψ ω: XGBoost for event selection [slides]

Search for lepton flavor-violating decay modes B0→K*τ±ℓ∓ (ℓ=e,μ) : fastBDT for qq and BB suppression

Measurement of the CP asymmetry in D+→ π+π0 decays: ROOT TMVA's implementation of MLP

Search for inclusive B →Xsvv decays using hadronic B tagging: fastBDT for event selection

Measurement of the time-integrated CP asymmetry in D0 →π0π0 decays: Histogram Based BDT for π0 selection

Measurement of the time-integrated CP asymmetry in D0 → KS KS decays: ROOT TMVA's BDT

How to do machine learning?

With a focus on supervised learning

Another summary from Student Research Showcase: [link] - slide 5

0 Data Collection & Processing

Get a dataset and process the data to prepare for training

1 Model Selection

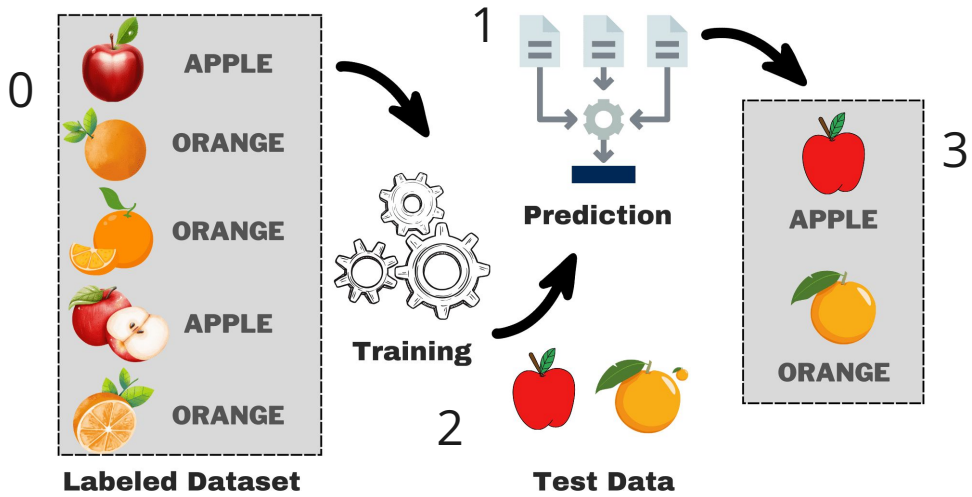
Choose the best set of algorithms that best suits your problem of interest

2 Training, Validation, & Testing

Given a dataset, tune your model to best solve the problem of interest.

3 Deployment

With your model trained and tested, all that's left to do is to apply it to new data with the trained parameters



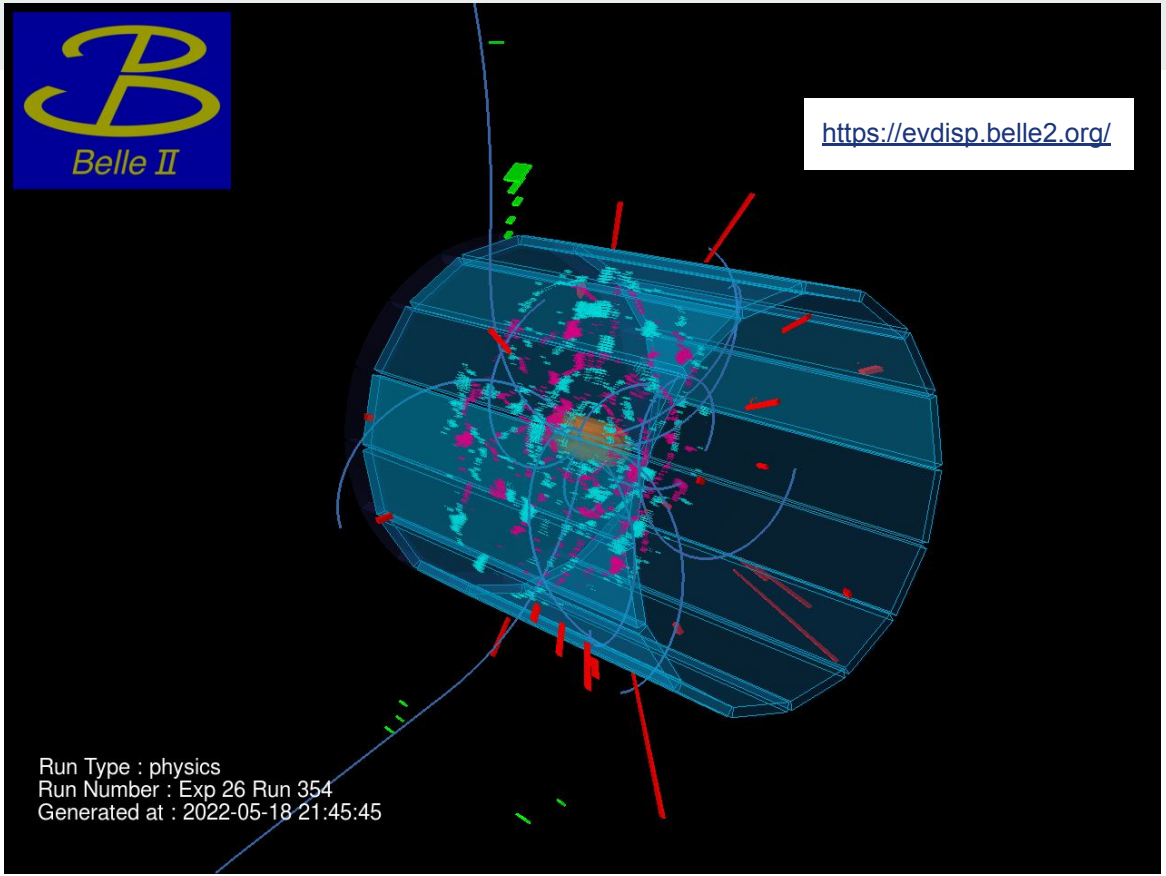
Data Collection

For Belle II, data taking is the most important task!

In general, the more data, the better.

Take shifts to help in with this part!

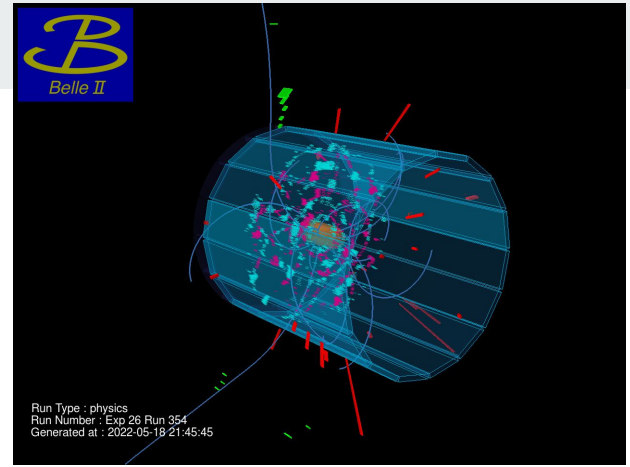
(shift2.belle2.org)

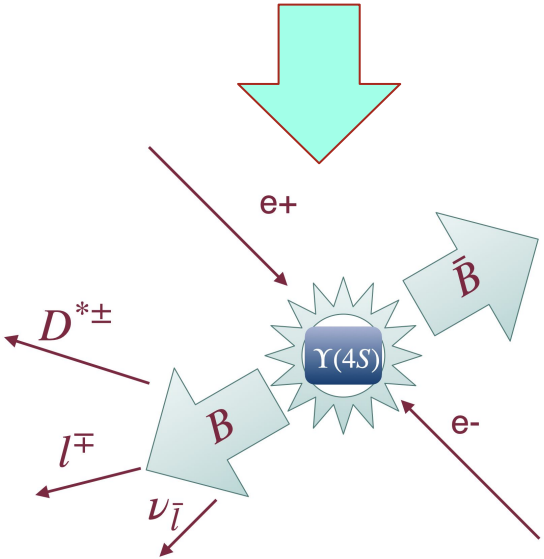


Data Processing

Basf2 software converts raw data to analysis-level variables as part of processing.

These variables are used as “**features**” or inputs into our machine learning model.

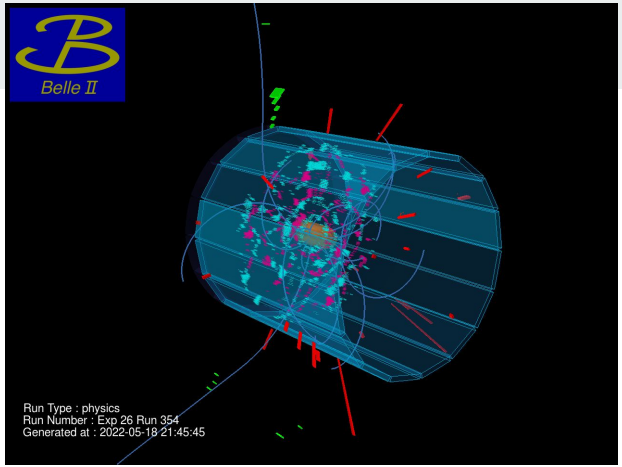


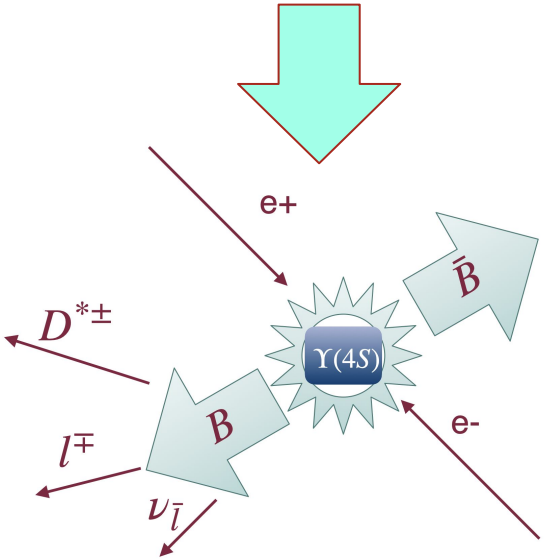


Data Processing

In general, data processing should also consider the following:

- Feature selection:
  - Choose variables with high discriminating power
  - Remove irrelevant variables
    - Little discriminating power
    - Highly correlated with other features
- Data Cleaning:
  - Missing Values/Bad Formatting/Type Conversion
  - Duplicates in dataset
  - Outlier detection (bad reconstructed signal?)
  - Verifying/cross-checking labels





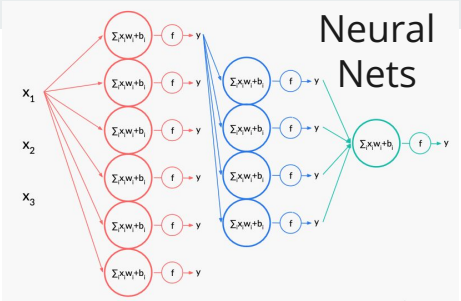
Model Selection

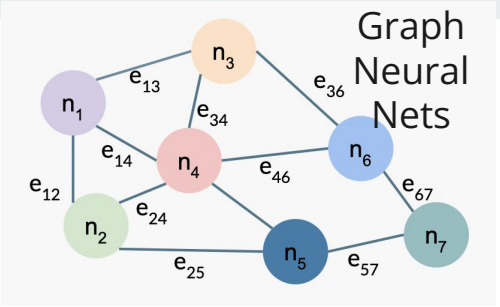
After preparing your datasets, choose the model/algorithm that is best suited for your task.

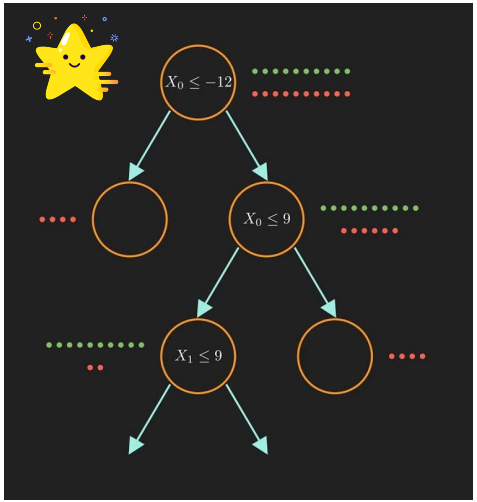
Things to consider:

- Availability of resources (time)
- Strengths and weaknesses?
- Evaluation Metrics

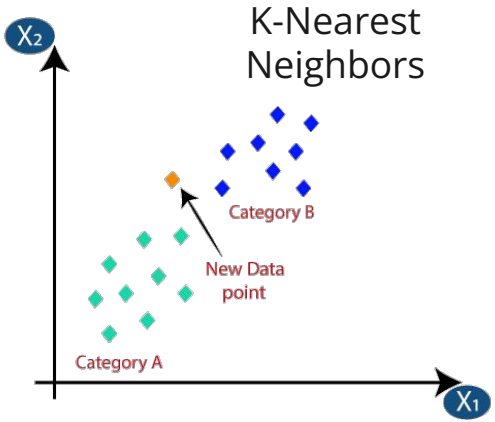
For details on models, see last year's summer workshop [link]







Decision Trees



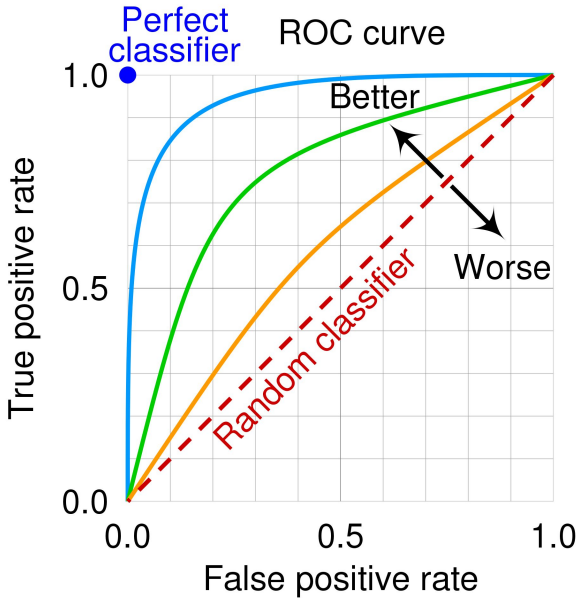
https://machinelearningmastery.com/applied-machine-learning-is-hard/

Training, Validation, & Testing

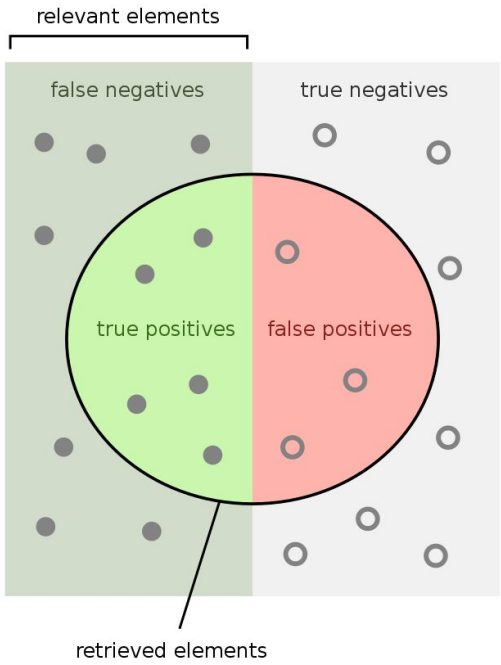
- For a given dataset with a set of features, one typically partitions their dataset into:
  - **Training**: The bulk of your overall dataset should be here to make sure your model learns as many trends as possible.
  - **Validation**: An independent subset used to check for overfitting during training (and hyper-parameter tuning)
  - **Testing**: Another independent subset used for overall comparisons.
- A figure of merit or evaluation metric is important and is dependent on your model and problem.

Performance Evaluation Example

Receiver Operating Characteristic (**ROC**) curves are used to evaluate classification models.



PC: https://en.wikipedia.org/wiki/Receiver_operating_characteristic



How many retrieved items are relevant?

Precision =

How many relevant items are retrieved?

Recall =

Quick Break For Questions

Understanding model

Decision Trees

0 **Data Collection & Processing**

Get a dataset and process the data to prepare for training

1 **Model Selection**

Choose the best set of algorithms that best suits your problem of interest

2 **Training, Validation, & Testing**

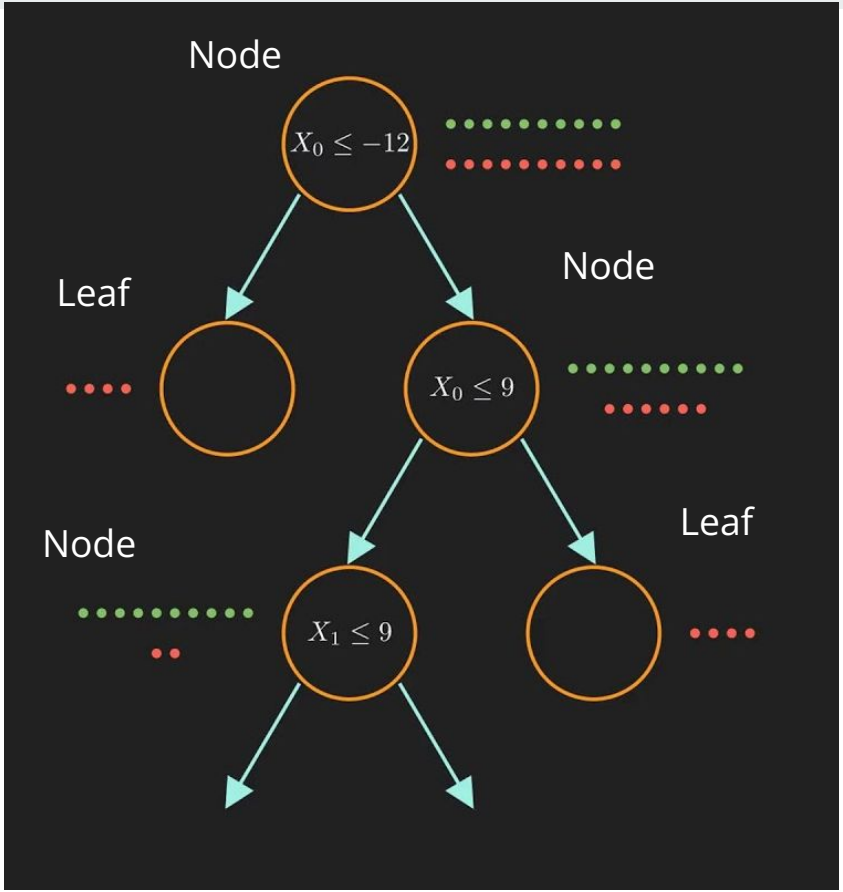
Given a dataset, tune your model to best solve the problem of interest.

3 **Deployment**

With your model trained and tested, all that's left to do is to apply it to new data with the trained parameters

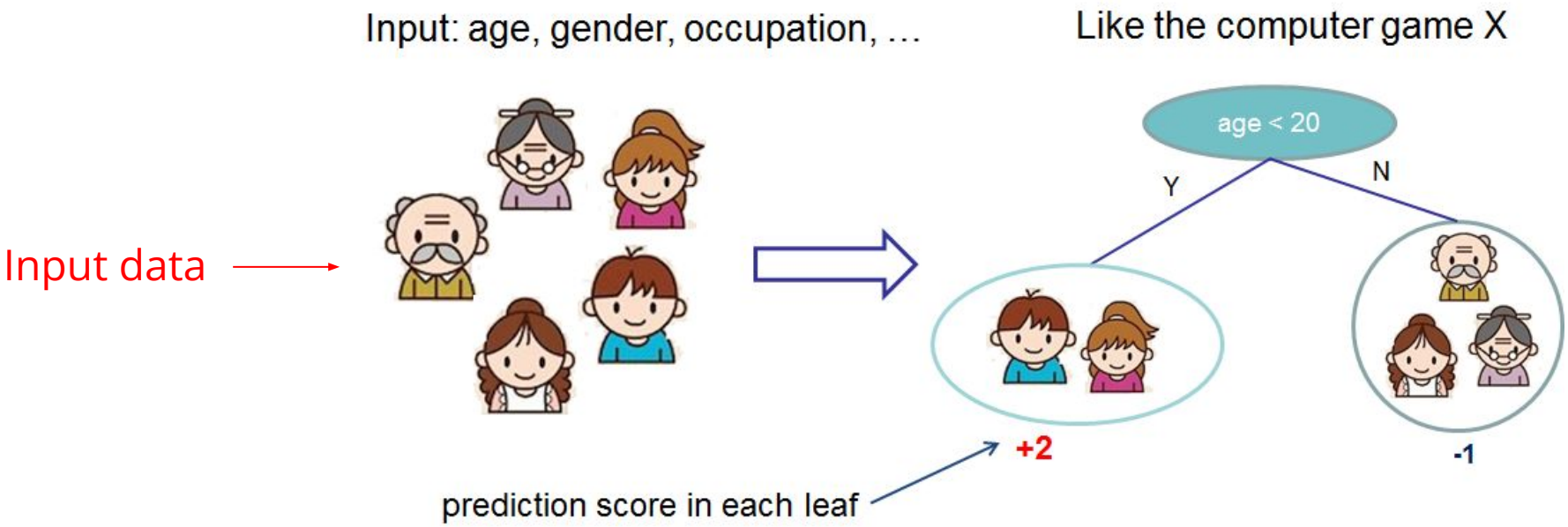
What is a decision tree?

- A decision tree is a flowchart-like model
- Each node partition the dataset based on one feature.
- Each leaf represents the outcome of the partition.



Example: Like vs. Dislike computer game

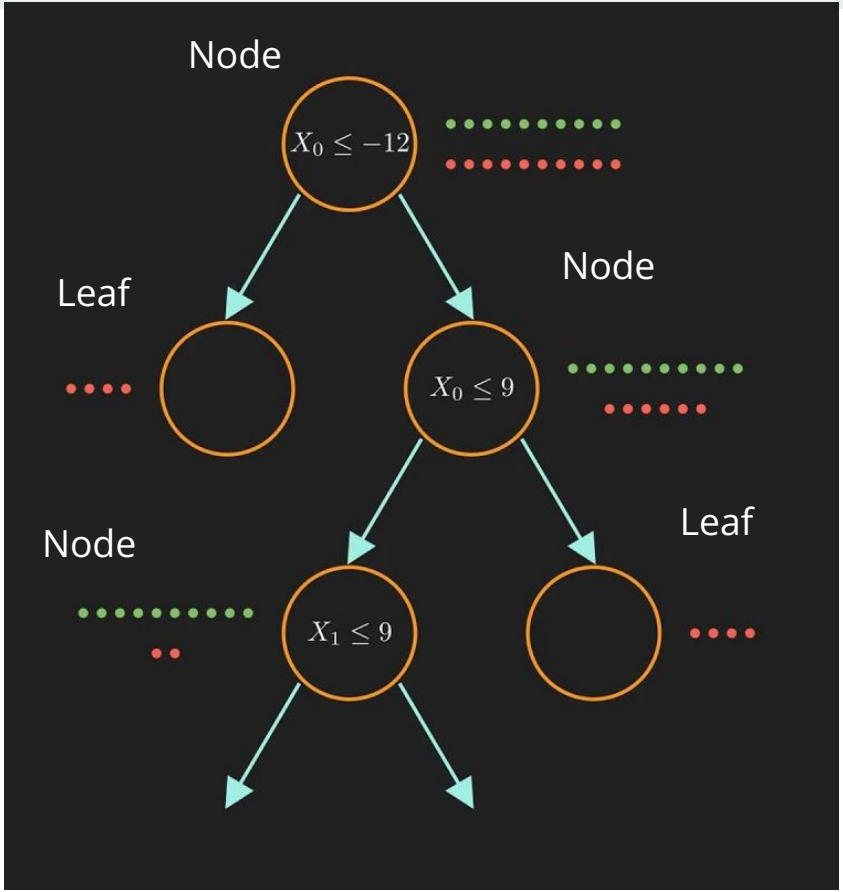
Decision Tree Model



Credit: Gradient Boost Intro

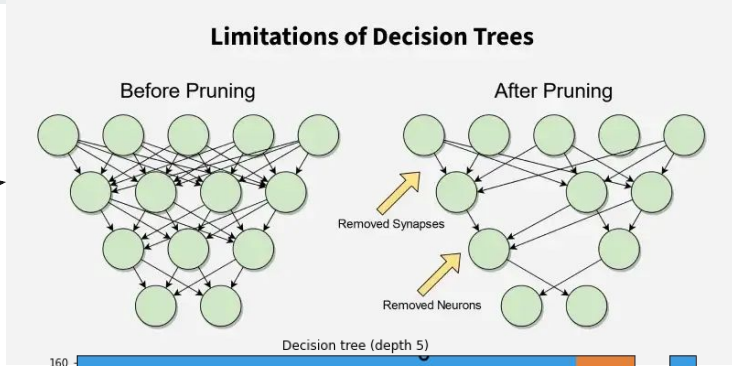
How decisions are made?

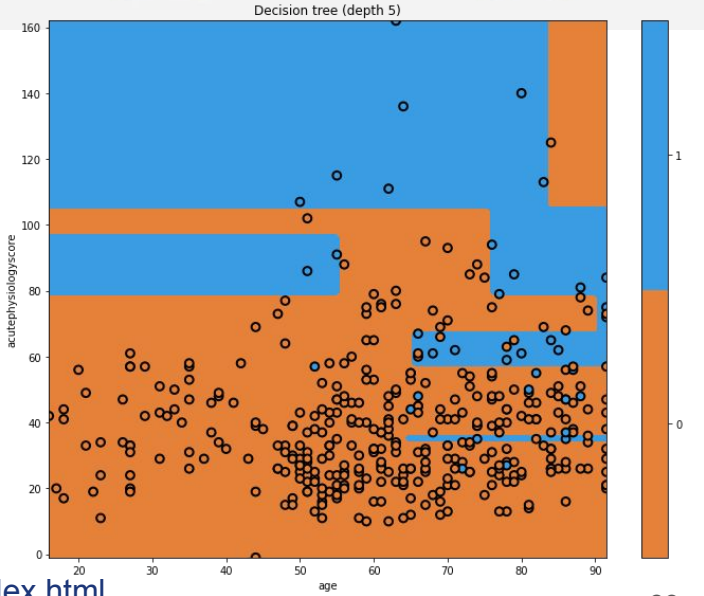
- The decision is made by minimizing the Gini Impurity or entropy of the subset
  - Gini = $\sum_i(p_i)(1-p_i) = 1 - \sum_i(p_i)^2$
  - Entropy = $-\sum_i(p_i)\log(p_i)$
  - Others…
- All possible splits are considered and optimal cut is selected



Limitations w/ Decision Trees

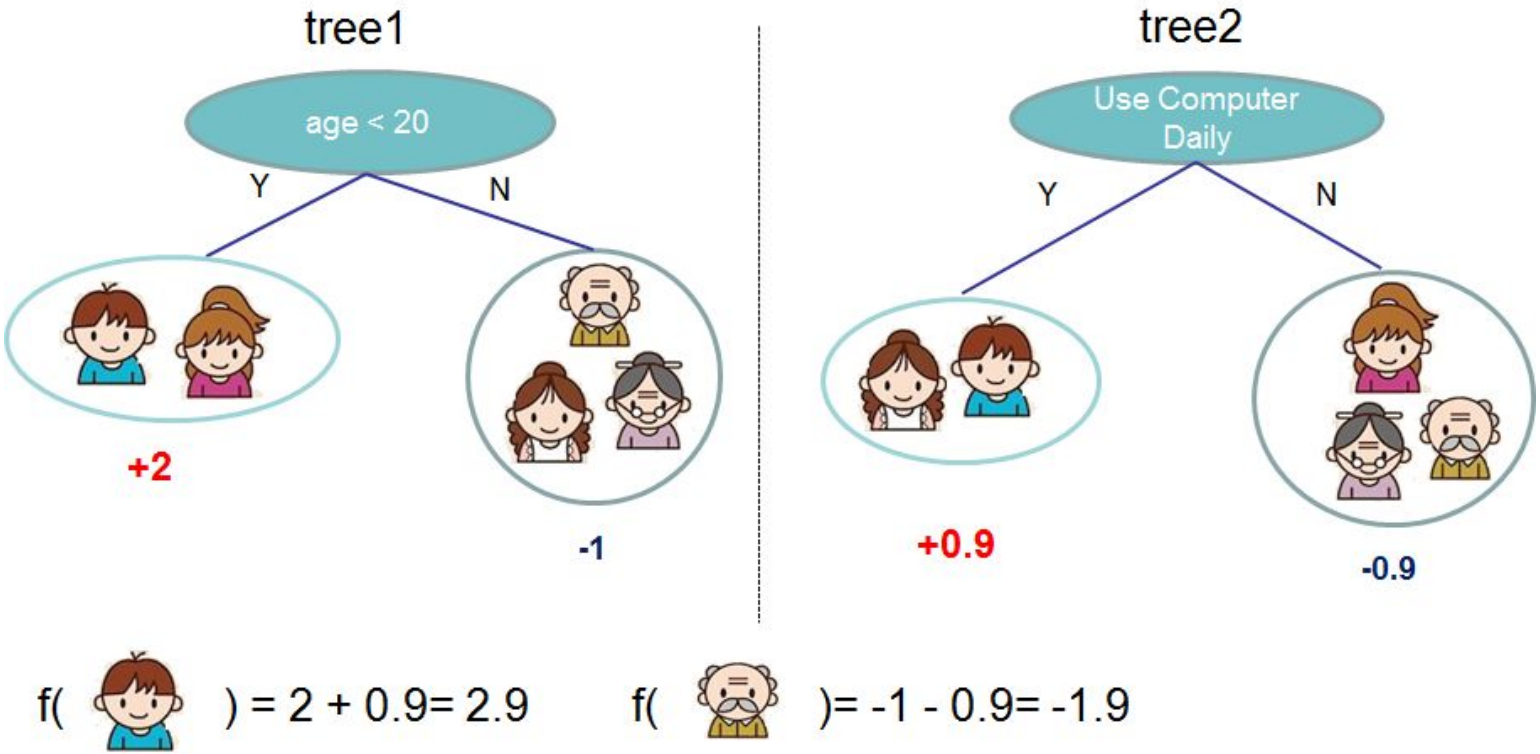
- Overfitting…
- High Variance
- Greedy Algorithm (makes locally optimal decisions)
- Bias towards dominate class (assuming there's a class imbalance)





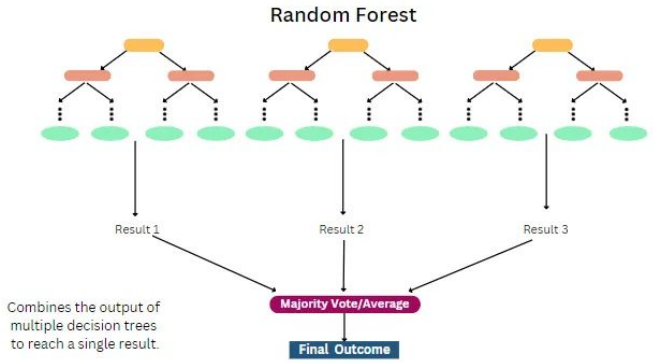
https://www.geeksforgeeks.org/limitations-of-decision-tree/,
https://carpentries-incubator.github.io/machine-learning-trees-python/03-variance/index.html

Decision Tree Ensembles

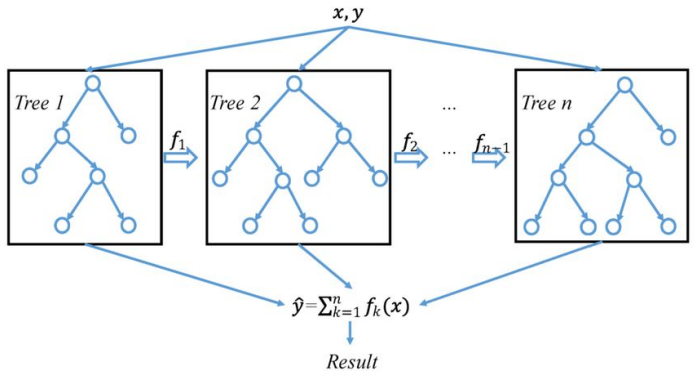


Split nodes **Grow trees** **Combine trees** **Learn!**

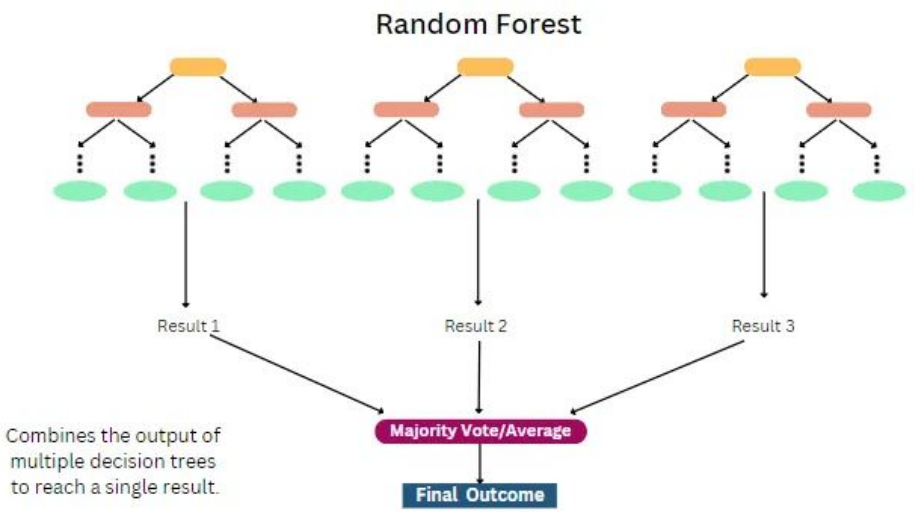
Random Forest



Boosted Decision Trees



Random Forests

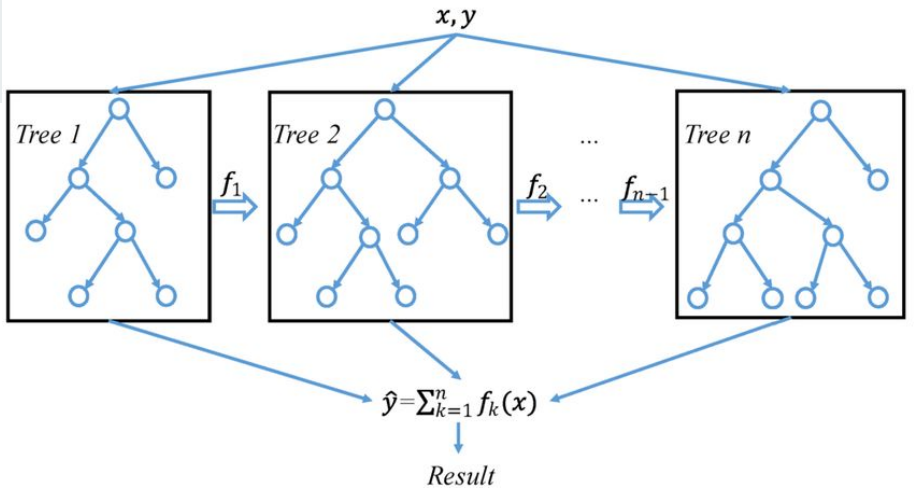


- Use an ensemble of models and classify an event by the majority rule
  - Concept of ‘bagging’ or ‘bootstrapping’
- Less susceptible to fluctuations and helpful for determining how important variables are

- A bit computationally expensive and less interpretable…

https://www.ibm.com/think/topics/random-forest

(Gradient-) Boosted DTs



For n epochs, train a DT and use information from said tree to inform structures of subsequent trees

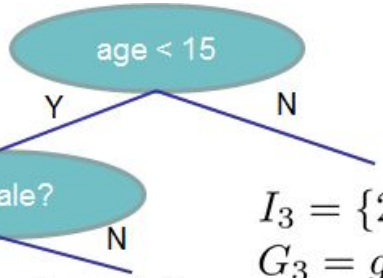
- Low bias, very easy implementation

- Can still prone to overfitting, so proceed with caution (i.e. hyperparameter tune)
- Still an intense computation (but not as much as random forest)

https://www.ibm.com/think/topics/boosting

Boosted Decision Trees (example)

Instance index	gradient statistics
1 	g1, h1
2 	g2, h2
3 	g3, h3
4 	g4, h4
5 	g5, h5



$$I_1 = \{1\} \quad I_2 = \{4\} \quad I_3 = \{2, 3, 5\}$$
$$G_1 = g_1 \quad G_2 = g_4 \quad G_3 = g_2 + g_3 + g_5$$
$$H_1 = h_1 \quad H_4 = h_4 \quad H_3 = h_2 + h_3 + h_5$$

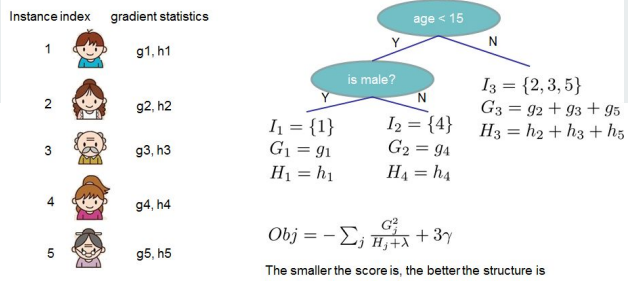
$$Obj = -\sum_j \frac{G_j^2}{H_j + \lambda} + 3\gamma$$

The smaller the score is, the better the structure is

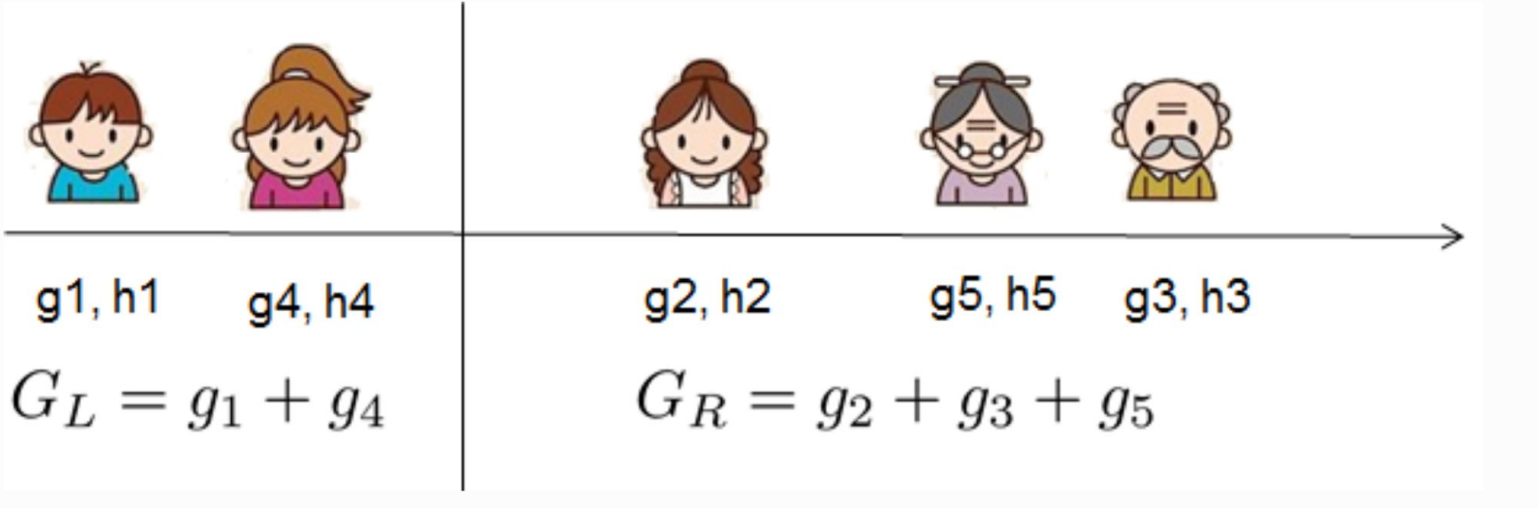
$$g_i = \partial_{\hat{y}_i^{(t-1)}} l(y_i, \hat{y}_i^{(t-1)})$$
$$h_i = \partial^2_{\hat{y}_i^{(t-1)}} l(y_i, \hat{y}_i^{(t-1)})$$

$$\text{obj}^{(t)} = \sum_{i=1}^{n} l(y_i, \hat{y}_i^{(t)}) + \sum_{k=1}^{t} \omega(f_k)$$

https://xgboost.readthedocs.io/en/stable/tutorials/model.html



Boosted Decision Trees (example)

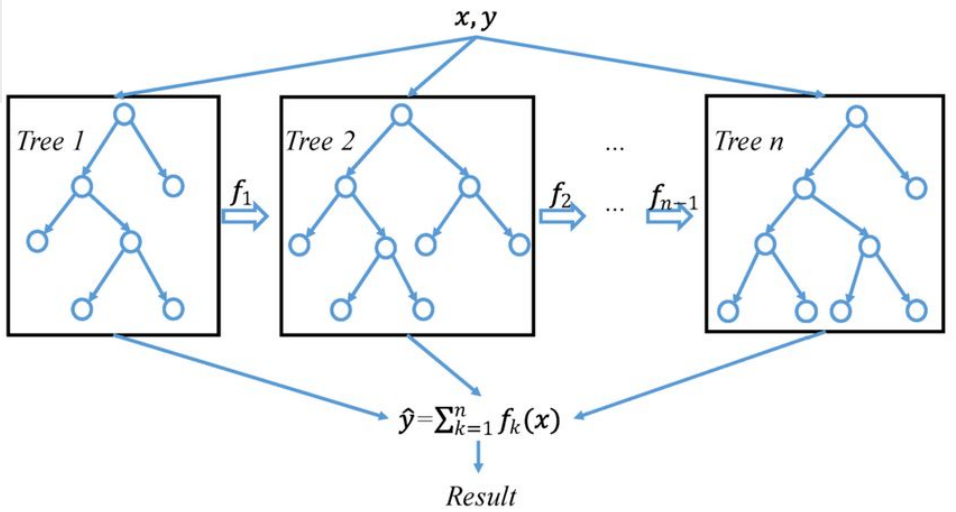


$$g_i = \partial_{\hat{y}_i^{(t-1)}} l(y_i, \hat{y}_i^{(t-1)})$$
$$h_i = \partial^2_{\hat{y}_i^{(t-1)}} l(y_i, \hat{y}_i^{(t-1)})$$

$$Gain = \frac{1}{2}\left[\frac{G_L^2}{H_L + \lambda} + \frac{G_R^2}{H_R + \lambda} - \frac{(G_L + G_R)^2}{H_L + H_R + \lambda}\right] - \gamma$$

https://xgboost.readthedocs.io/en/stable/tutorials/model.html

Packages



- XGBoost: https://xgboost.readthedocs.io/en/stable/
  - Expert(s): Noah Brenny, maybe me…
- FastBDT: https://software.belle2.org/development/sphinx/mva/doc/index-01-mva.html#fastbdt
  - Expert(s): Logan Benninghoff, Alex Gale
- LightGBM: https://lightgbm.readthedocs.io/en/latest/Features.html
  - Expert(s): Boyang Zhang
- Scikit-Learn: https://scikit-learn.org/stable/modules/ensemble.html
  - Expert(s): Santi Naylor
- ROOT's TMVA: https://root.cern/manual/tmva/

How to use XGBoost: Hyper-parameters

- Boosting Parameters
  - Learning rate (α), minimization loss for new leaf (γ), max tree depth, minimum child weight (i.e. when to give up on partitioning), subsampling rate, subsampling method, regularization weights, …
- Learning Task Parameters
  - Objective function, evaluation metrics (for validation)

- How to choose… (hint: slide 13)

https://xgboost.readthedocs.io/en/stable/parameter.html

How to use XGBoost: Example Script

```
from xgboost import XGBClassifier
import uproot
import pandas as pd
from sklearn.model_selection import train_test_split

# read data
tree = uproot.open("path/to/dataset.root:myTree")
data = tree.arrays(library='pd')
X_train, X_test, y_train, y_test = train_test_split(data['data'],
                                                    data['target'], test_size=.2)

# create model instance
random_parameters = {'learning_rate': 0.1, 'max_depth': 3,
                     'n_estimators': 42, 'objective':'binary:logistic'}
bst = XGBClassifier(**random_parameters)

# fit model
bst.fit(X_train, y_train)

# make predictions
preds = bst.predict(X_test)
```

https://xgboost.readthedocs.io/en/latest/parameter.html

Also see:
https://www.kaggle.com/code/zjkjsd/xgboost-example

How to use scikit-learn: Example Script

```
from sklearn.ensemble import GradientBoostingClassifier
import uproot
import pandas as pd
from sklearn.model_selection import train_test_split

# read data
tree = uproot.open("path/to/dataset.root:myTree")
data = tree.arrays(library='pd')
X_train, X_test, y_train, y_test = train_test_split(data['data'],
                                                    data['target'], test_size=.2)

# create model instance
random_parameters = {'learning_rate': 0.1, 'max_depth': 3,
                     'n_estimators': 42, 'loss':'log_loss'}

clf = GradientBoostingClassifier(**random_parameters)


# fit model
clf.fit(X_train, y_train)

# make predictions
preds = clf.predict(X_test)
```

https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingClassifier.html

Also see:
https://www.kaggle.com/code/zjkjsd/scikit-learn-example

How to use LightGBM: Example Script

```
import optuna # for hyperparameter tuning
import lightgbm as lgm
#import optuna.integration.lightgbm as lgb # using optuna intergraiont with lightgbm
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
```

```
param = {
    "objective": "binary",
    "metric": "binary_logloss",
    "verbosity": -1,
    "boosting_type": "gbdt",
    "lambda_l1": trial.suggest_float("lambda_l1", 1e-8, 10.0, log=True),
    "lambda_l2": trial.suggest_float("lambda_l2", 1e-8, 10.0, log=True),
    "num_leaves": trial.suggest_int("num_leaves", 2, 22),
    "feature_fraction": trial.suggest_float("feature_fraction", 0.4, 1.0),
    "bagging_fraction": trial.suggest_float("bagging_fraction", 0.4, 1.0),
    "bagging_freq": trial.suggest_int("bagging_freq", 1, 7),
    "min_child_samples": trial.suggest_int("min_child_samples", 5, 10),
}

gbm = lgm.train(param, dtrain)
preds = gbm.predict(X_val)
pred_labels = np.rint(preds)
```

https://lightgbm.readthedocs.io/en/stable/

Also see:
https://www.kaggle.com/code/zjkjsd/lightgbm-example

How to use FastBDT: Example Script

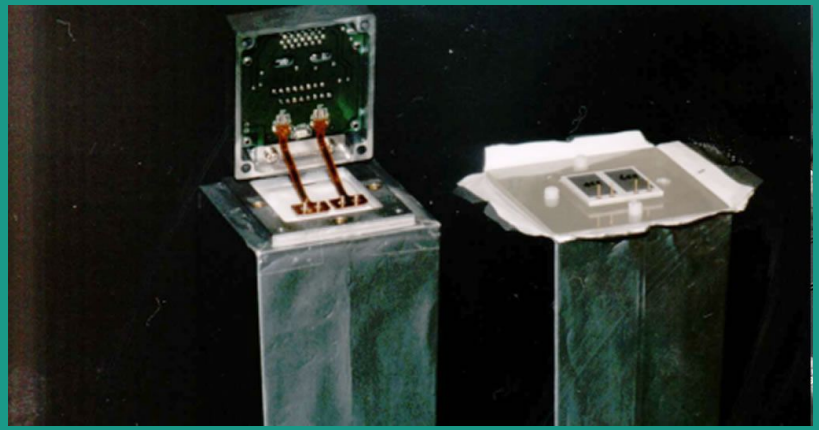
```
from PyFastBDT import FastBDT
import uproot
import pandas as pd
from sklearn.model_selection import train_test_split

# read data
tree = uproot.open("path/to/dataset.root:myTree")
data = tree.arrays(library='pd')
X_train, X_test, y_train, y_test = train_test_split(data['data'],
                                                    data['target'], test_size=.2)

# create model instance
random_parameters = {"nTrees":1, "depth":1, "shrinkage":0.1, "subsample":1.0,}

# Train FastBDT using its PythonInterface,
# which is based on the SKLearn classifiers
clf = FastBDT.Classifier(**random_parameters)

# fit model
clf.fit(X=X_train, y=y_train)

# make predictions
preds = clf.predict(X_test)
```

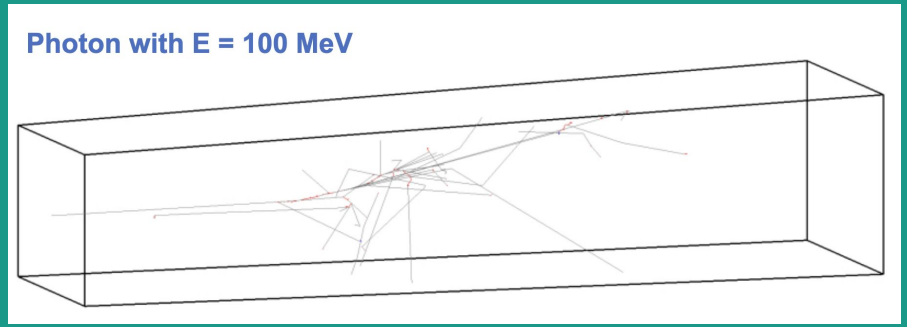
https://github.com/thomaskeck/FastBDT/tree/master

(No example made on Kaggle yet…)

Quick Break For Questions

Photon Detection Recap



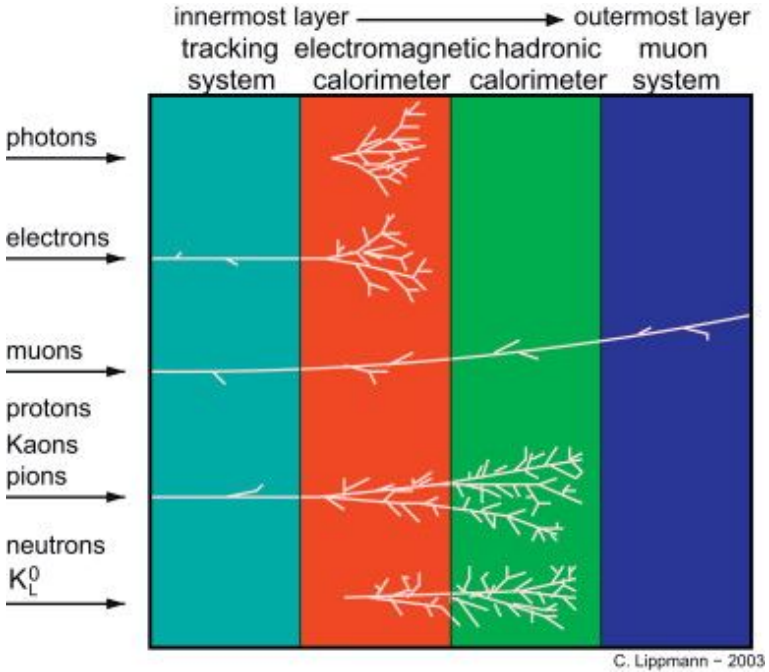


For details: https://indico.belle2.org/event/8841/contributions/61147/attachments/23523/34739/Calorimetry.pdf
Or https://indico.belle2.org/event/14599/contributions/93876/attachments/35194/52138/B2SS2025_Detector.pdf (slides 19 & 20)

QUIZ:

- How do we detect photons?

How do we measure photons?



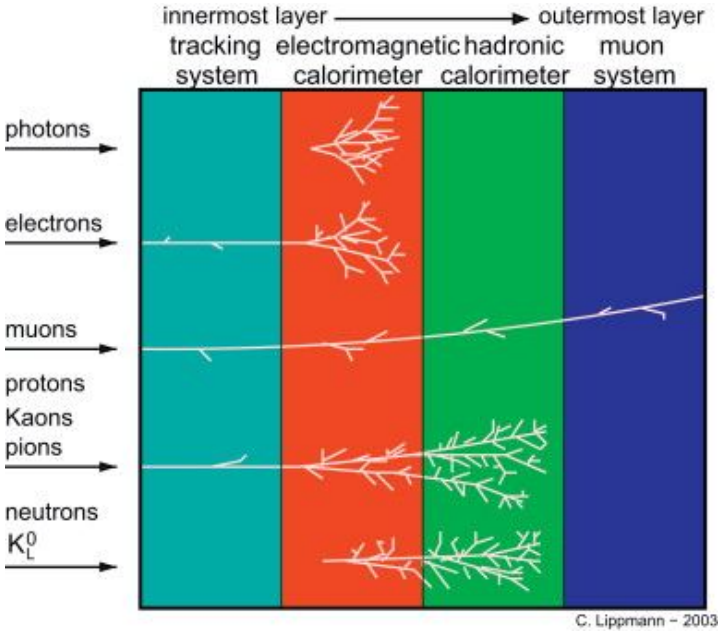
- Electromagnetic Calorimeter (ECL) is our hero!
- No tracks (i.e. isolated cluster)

QUIZ:

- How do we detect photons?
- Are there any FSPs that could fake a photon signal?

FSP = final state particles

What could fake photons?



- n, K-Longs, but not part of this discussion (how do we deal with them…)
- Multiple low-energy clusters from beam background processes
- Showers from tracks?

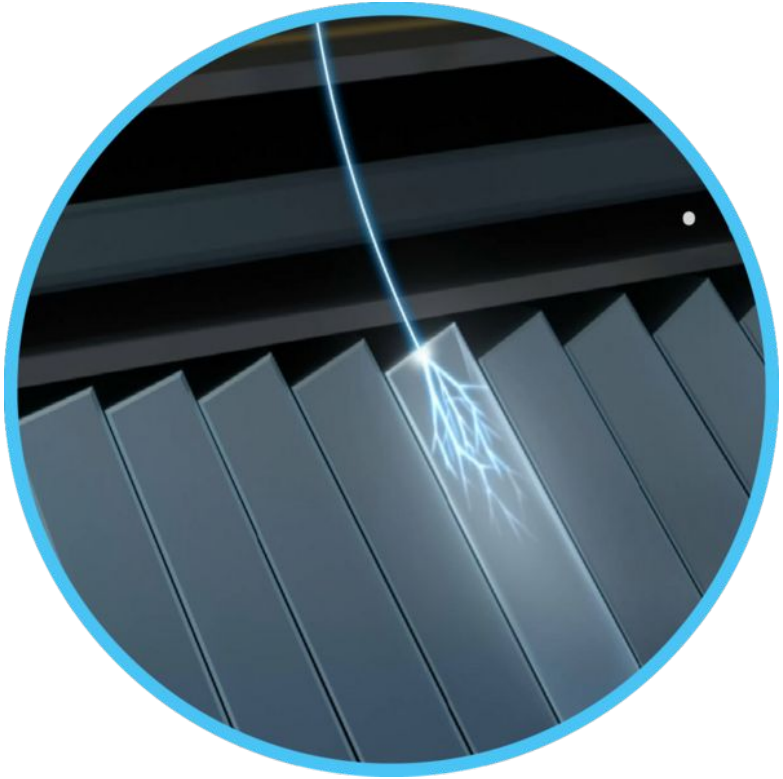
QUIZ:

- How do we detect photons?
- Are there any FSPs that could fake a photon signal?
- Are there any properties of ECL that could help us distinguish between photons and other FSPs?

FSP = final state particles

Cluster properties

- Measure the **energy deposition and the width of the shower**, e.g. E/p
- Electrons will create EM showers and deposit all their energy in the ECL
- Hadrons will likely pass through and not lose much energy (so its shape would be different than EM showers)

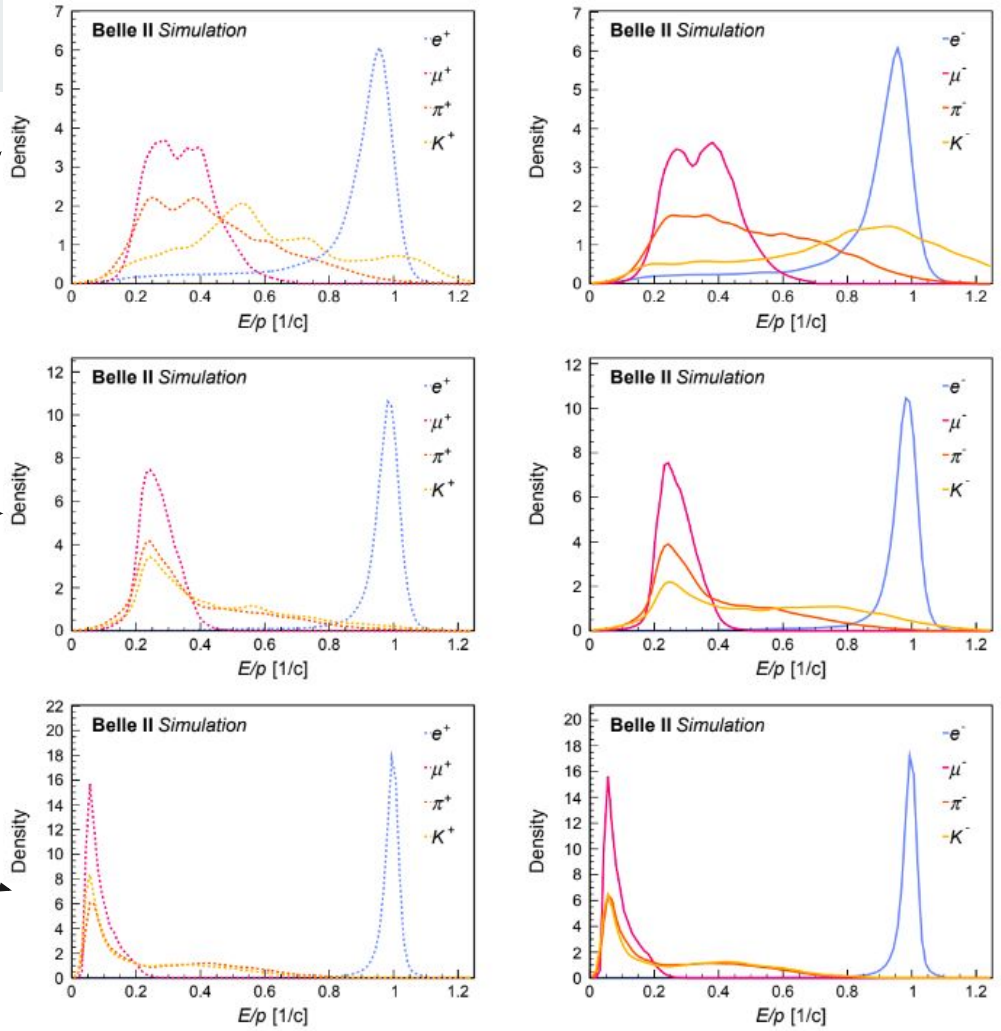


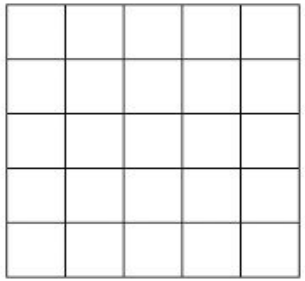
ECL and E/p:

0.2< p < 0.6 GeV

0.6< p < 1 GeV

p > 1 GeV
Most powerful region
Compensate CDC





Potential variables to work with

- clusterE
- clusterEoP
- minC2TDist
- clusterE1E9
- clusterE9E25 (or E9E21)
- clusterAbsZernikeMoment40
- clusterTheta, clusterPhi
- clusterTiming,
- clusterErrorTiming
- clusterNHits
- clusterReg
- MVA outputs?

Full list: https://software.belle2.org/development/sphinx/analysis/doc/Variables.html#ecl-cluster

Beam Background Suppression

Much thanks to P. Cheema for her assistance with this hands-on!

Hands-on exercise: kaggle

https://www.kaggle.com/competitions/2025-b2sw-ml

Example notebooks:

https://www.kaggle.com/competitions/2025-b2sw-ml/code

Backup

Regularization

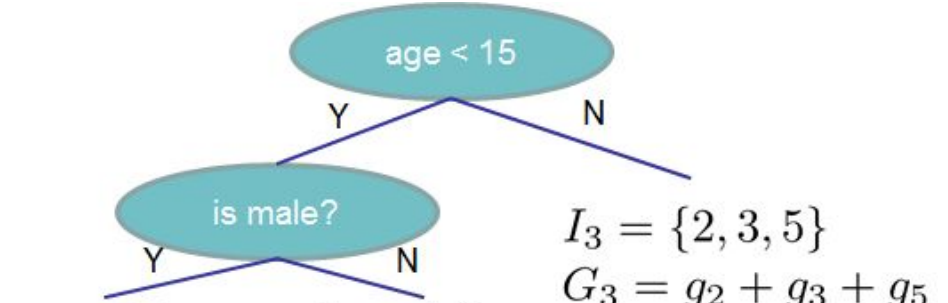
ex. $\mathbb{L} = \mathscr{L}^2 + \lambda \sum_n |\vec{w}_n|^2$

For certain models, one way to tackle overfitting and exploding gradients is by introducing a regularizing term in the loss function.

Sometimes, λ is normalized by the number of nodes/weights you have (but is up to the developer/user) since it is a constant throughout training anyways

New node split and new tree?

Instance index		gradient statistics
1		g1, h1
2		g2, h2
3		g3, h3
4		g4, h4
5		g5, h5



$$I_1 = \{1\} \quad I_2 = \{4\} \quad I_3 = \{2, 3, 5\}$$
$$G_1 = g_1 \quad G_2 = g_4 \quad G_3 = g_2 + g_3 + g_5$$
$$H_1 = h_1 \quad H_4 = h_4 \quad H_3 = h_2 + h_3 + h_5$$

$$Obj = -\sum_j \frac{G_j^2}{H_j + \lambda} + 3\gamma$$

The smaller the score is, the better the structure is

Algorithm 1: Gradient_Boost

1. $F_0(\mathbf{x}) = \arg\min_\rho \sum_{i=1}^{N} L(y_i, \rho)$
2. For $m = 1$ to M do:
3. $\quad \tilde{y}_i = -\left[\frac{\partial L(y_i, F(\mathbf{x}_i))}{\partial F(\mathbf{x}_i)}\right]_{F(\mathbf{x})=F_{m-1}(\mathbf{x})}, \; i = 1, N$
4. $\quad \mathbf{a}_m = \arg\min_{\mathbf{a},\beta} \sum_{i=1}^{N} [\tilde{y}_i - \beta h(\mathbf{x}_i; \mathbf{a})]^2$
5. $\quad \rho_m = \arg\min_\rho \sum_{i=1}^{N} L(y_i, F_{m-1}(\mathbf{x}_i) + \rho h(\mathbf{x}_i; \mathbf{a}_m))$
6. $\quad F_m(\mathbf{x}) = F_{m-1}(\mathbf{x}) + \rho_m h(\mathbf{x}; \mathbf{a}_m)$
7. endFor

end Algorithm

Algorithm 2: LS_Boost

$F_0(\mathbf{x}) = \bar{y}$

For $m = 1$ to M do:

$\quad \tilde{y}_i = y_i - F_{m-1}(\mathbf{x}_i), \quad i = 1, N$

$\quad (\rho_m, \mathbf{a}_m) = \arg\min_{\mathbf{a},\rho} \sum_{i=1}^{N} [\tilde{y}_i - \rho h(\mathbf{x}_i; \mathbf{a})]^2$

$\quad F_m(\mathbf{x}) = F_{m-1}(\mathbf{x}) + \rho_m h(\mathbf{x}; \mathbf{a}_m)$

endFor

end Algorithm

https://jerryfriedman.su.domains/ftp/trebst.pdf

Resources (in general)

In the last 5 years, there is a major abundance of resources to learn ML. Let put a subset of resources that I used to inspire this talk:

S. Still. “Machine Learning: For scientists” (2019).

I. Haide and L. Reuter “Machine Learning: Current Projects at Belle 2” (2023).

S. Dubey. “Machine Learning Hands-On” (2023).

S. Vallecorsa. “Artificial Intelligence and Machine Learning” CHEP 2023.

J. Hurwitz and D. Kirsch “Machine Learning for dummies: IBM Limited Edition.”

P. Wittek. “Quantum Machine Learning: What Quantum Computing Means to Data Mining” (2014).

Resources (Decision Trees)

- “Boosted Decision Trees.” https://arxiv.org/pdf/2206.09645
- “What is a decision tree?.” https://www.ibm.com/topics/decision-trees
- “Decision Tree.” https://en.wikipedia.org/wiki/Decision_tree
- “FastBDT: A speed-optimized and cache-friendly implementation of stochastic gradient-boosted decision trees for multivariate classification.” https://arxiv.org/abs/1609.06119
- “XGBoost: A Scalable Tree Boosting System.” https://arxiv.org/abs/1603.02754