Machine Learning Learning Machine

Machine Learning Basics

Yuri Mitrankov

Stony Brook University

December 05, 2024

Stony Brook University

Standard Model of ML running:

1 Inheriting ML Code

2 Modify

3 Create your training sample

4 Get total nonsense

5 Google for answers

6 Eureka moment

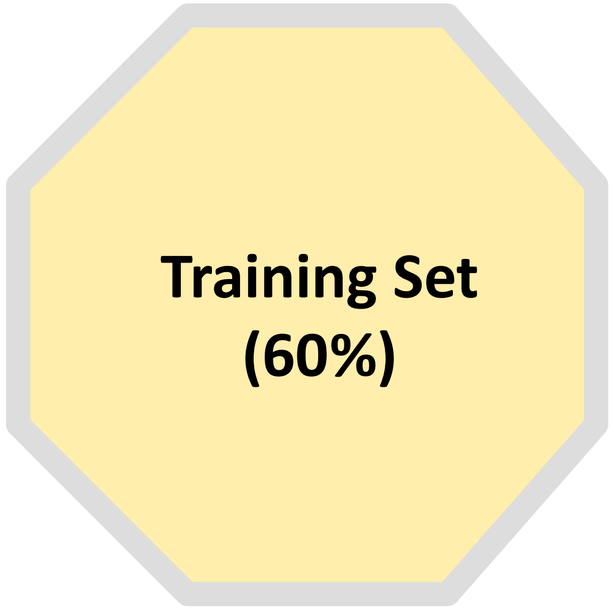
7 Try it

Repeat from step 5 until succeed

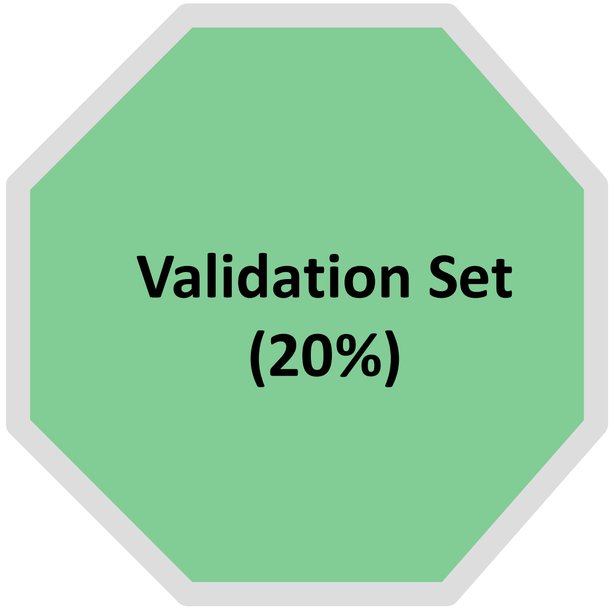
ML running basics

ML running basics

Stony Brook University



To train the models



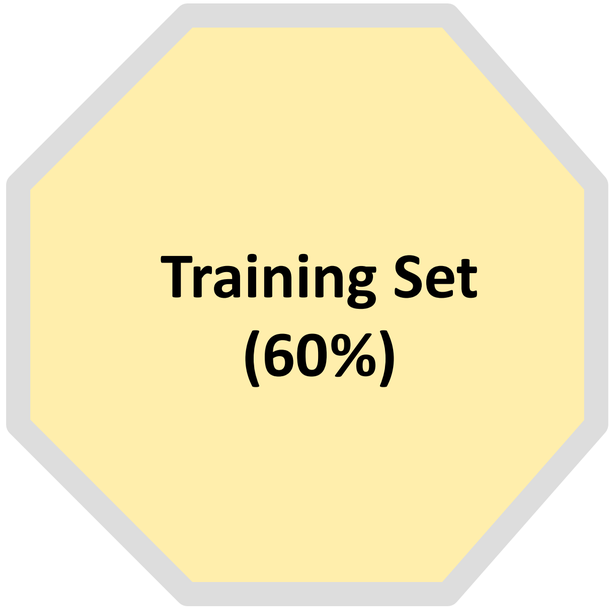
To prevent overfitting

To determine accuracy

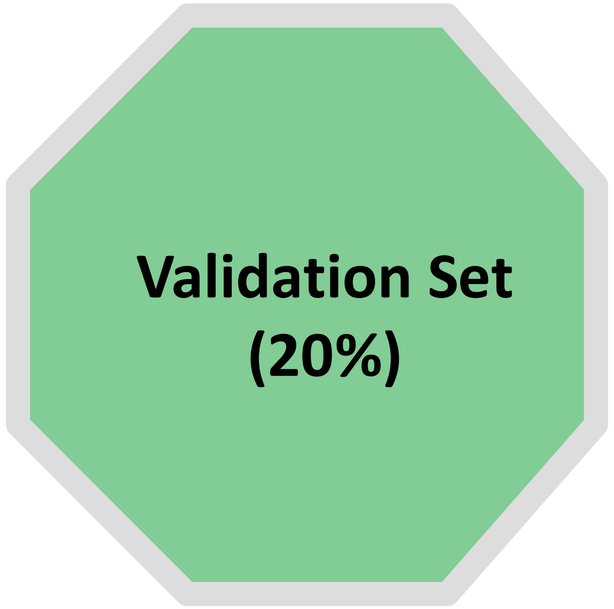
Each ML model has multiple parameters:
Run with various sets then find the best

ML running basics

Stony Brook University



To train the models



To prevent overfitting

To determine accuracy

Each ML model has multiple parameters:
Run with various sets then find the best
FOR YOU

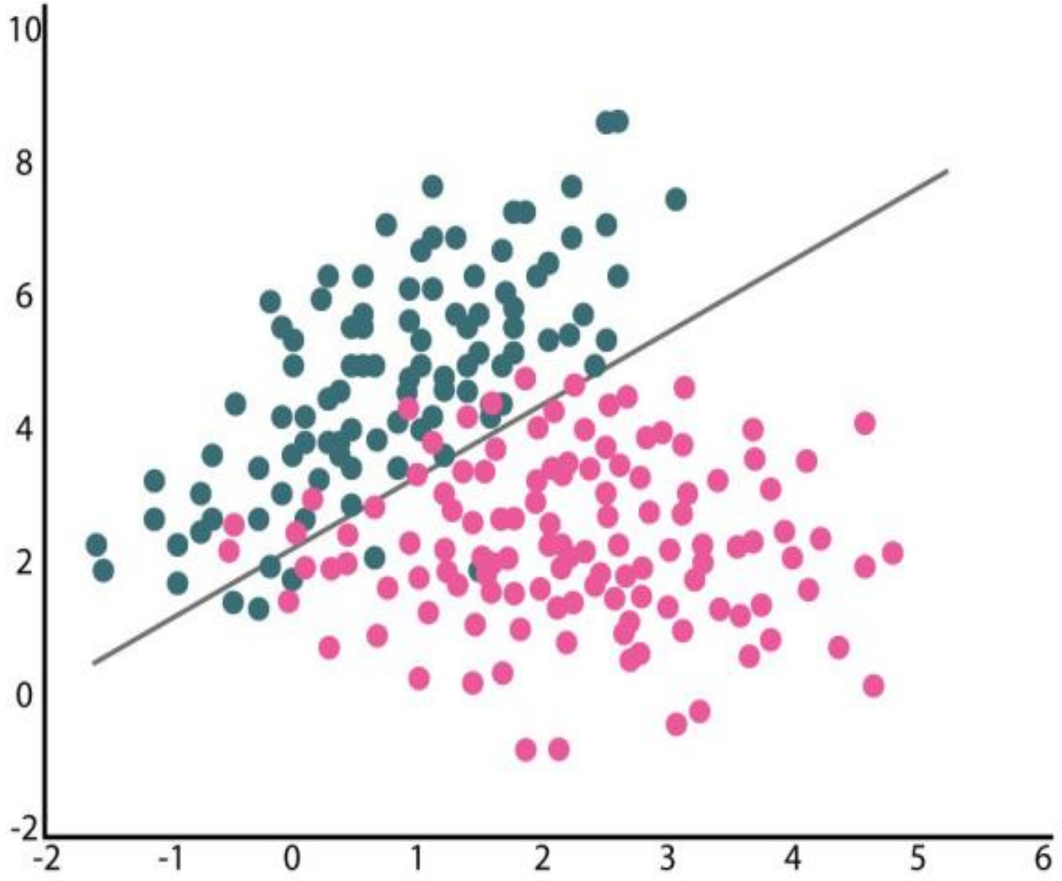
Stony Brook University

Classification in ML

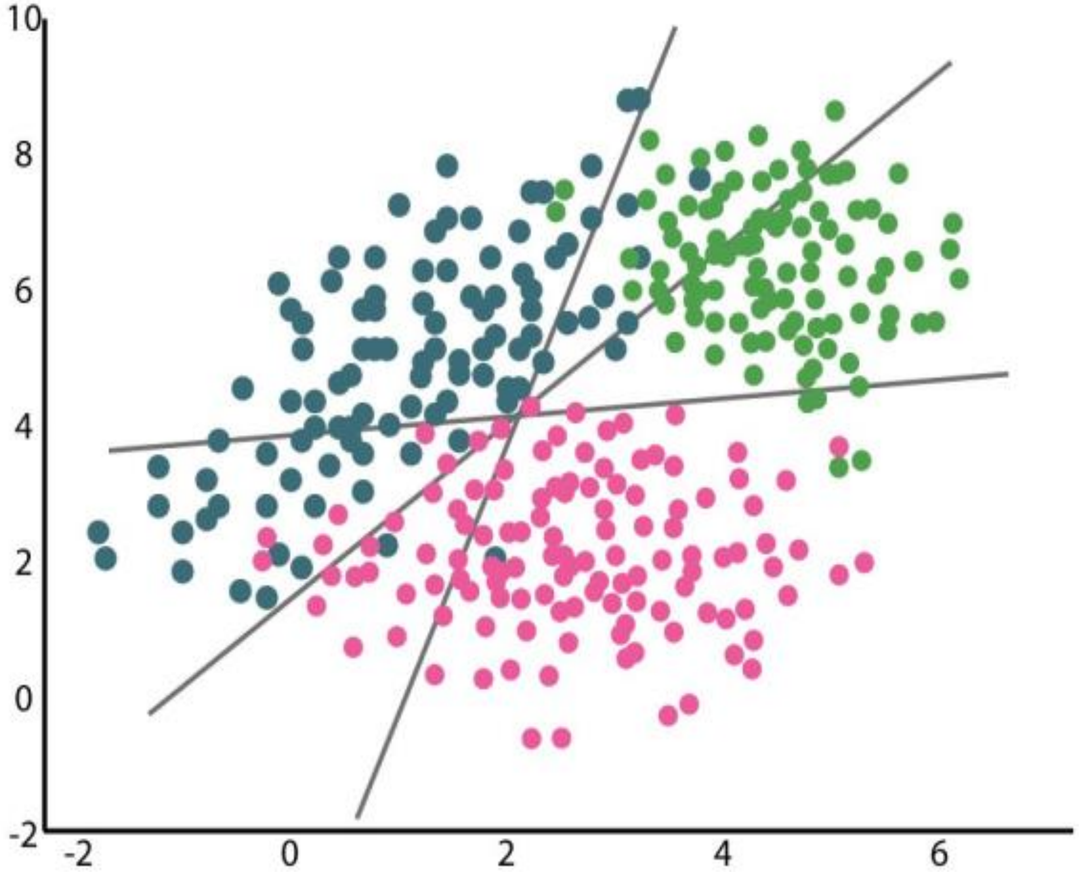
Classification in ML

Stony Brook University

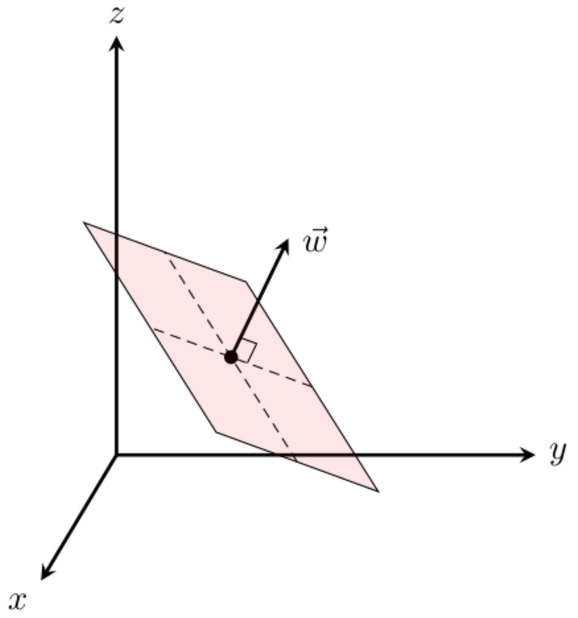
Binary classification

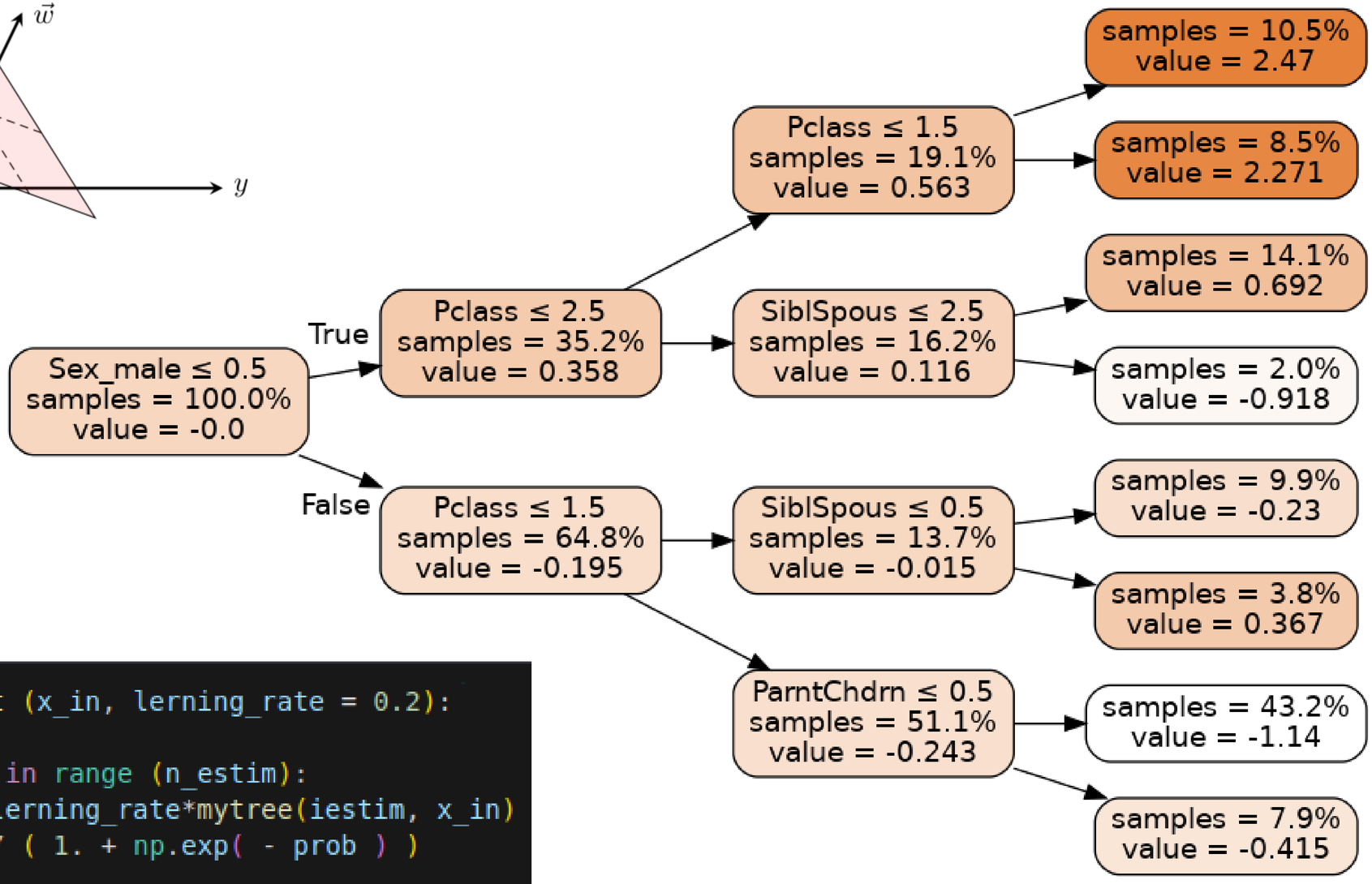


Multi-class classification



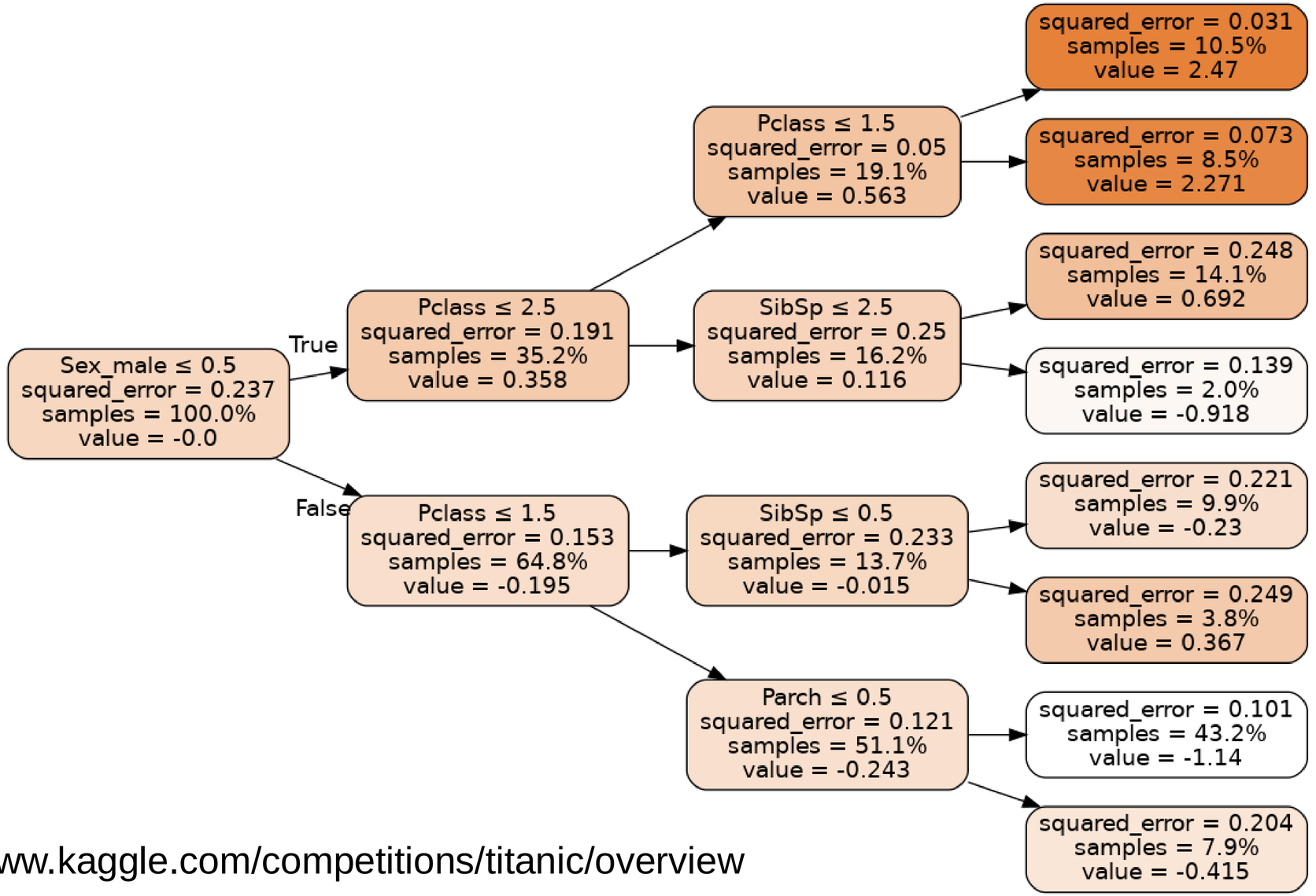
Example






```python
def mygradboost (x_in, lerning_rate = 0.2):
    prob = 0
    for iestim in range (n_estim):
        prob+=lerning_rate*mytree(iestim, x_in)
    return 1. / ( 1. + np.exp( - prob ) )
```

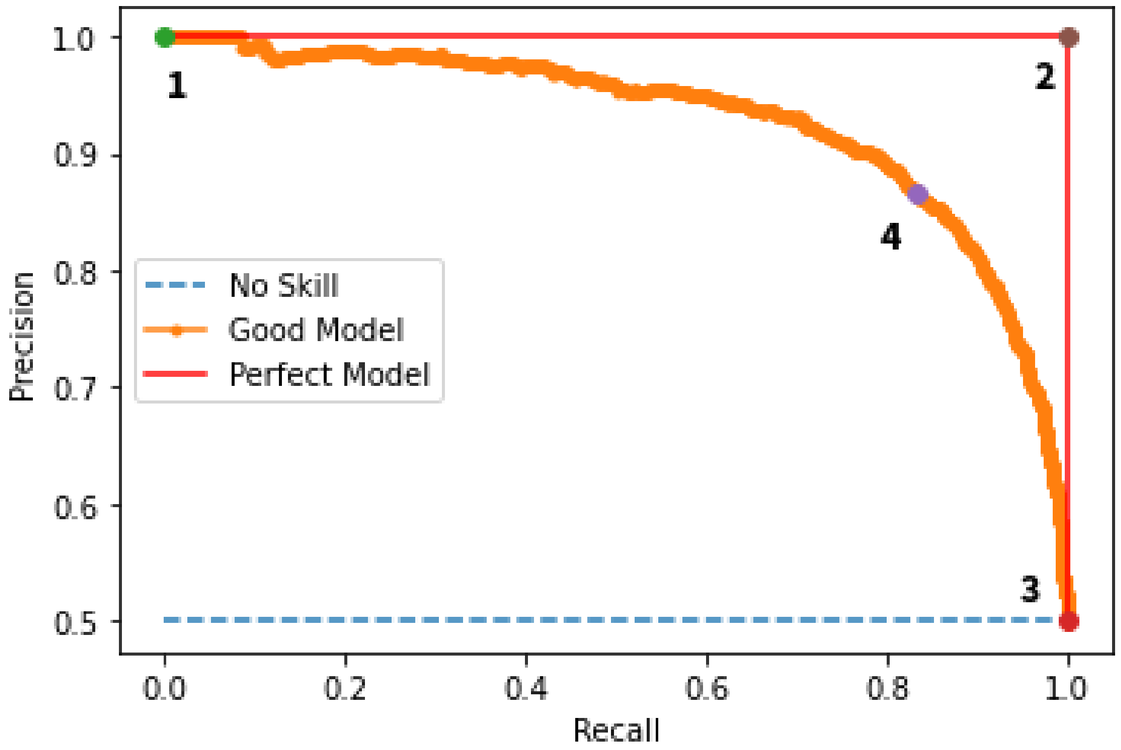
Example



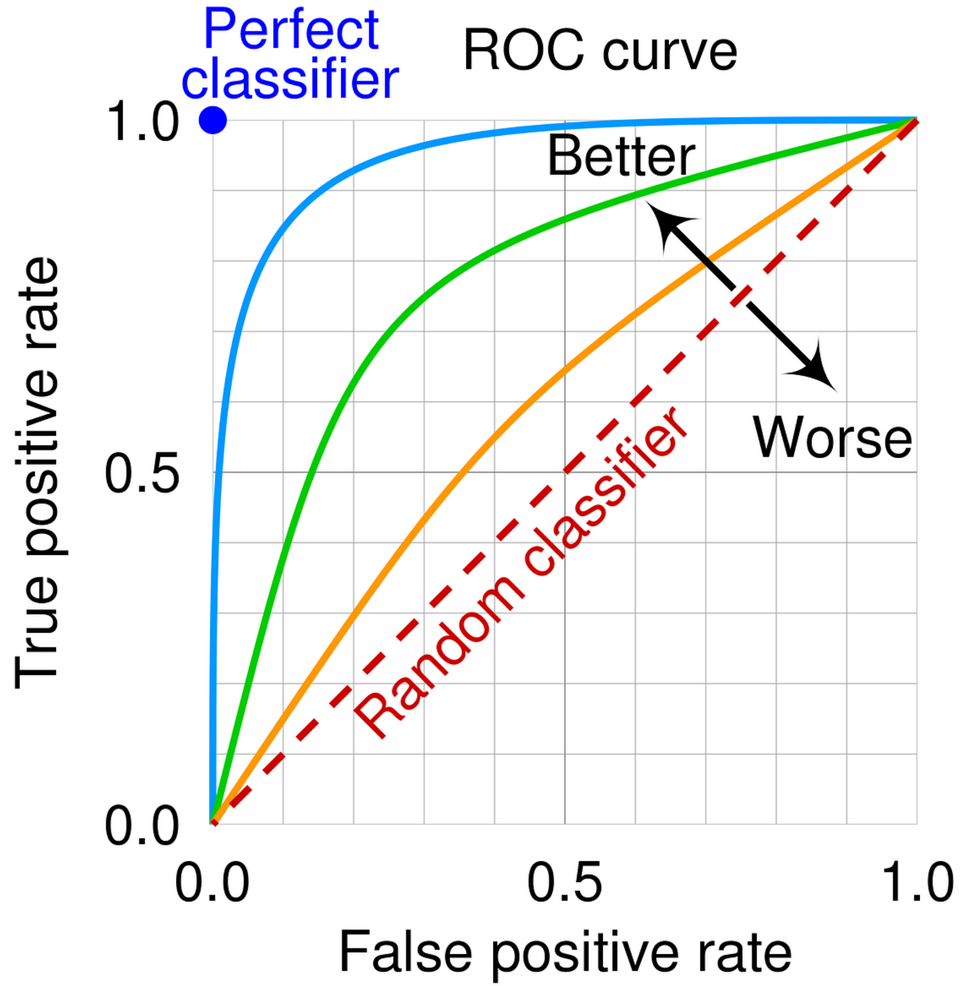
https://www.kaggle.com/competitions/titanic/overview

Quality metrics

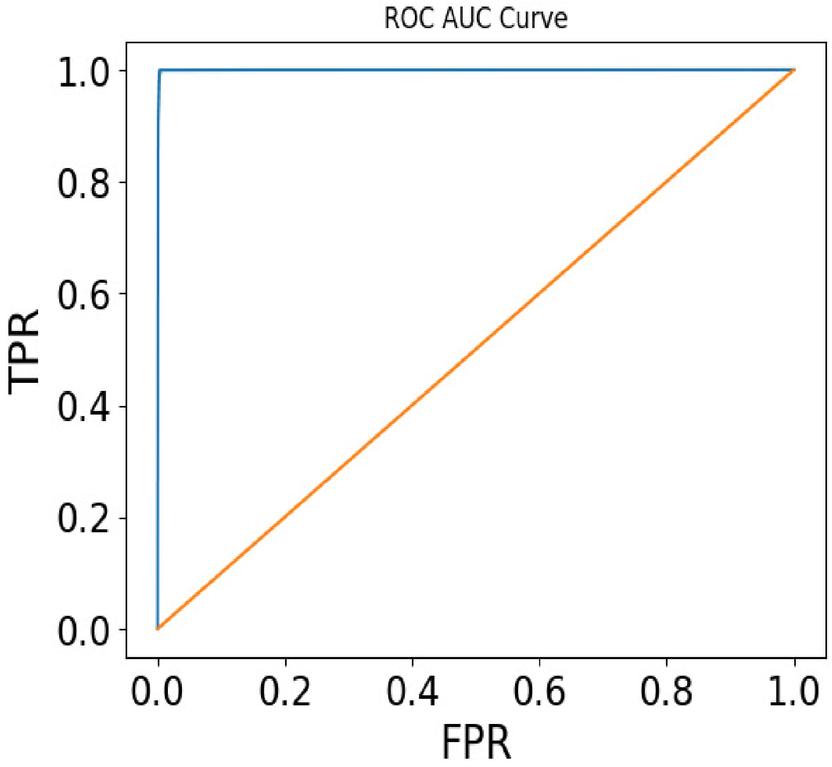
$$precision = \frac{TP}{TP + FP} \qquad recall = \frac{TP}{TP + FN}$$

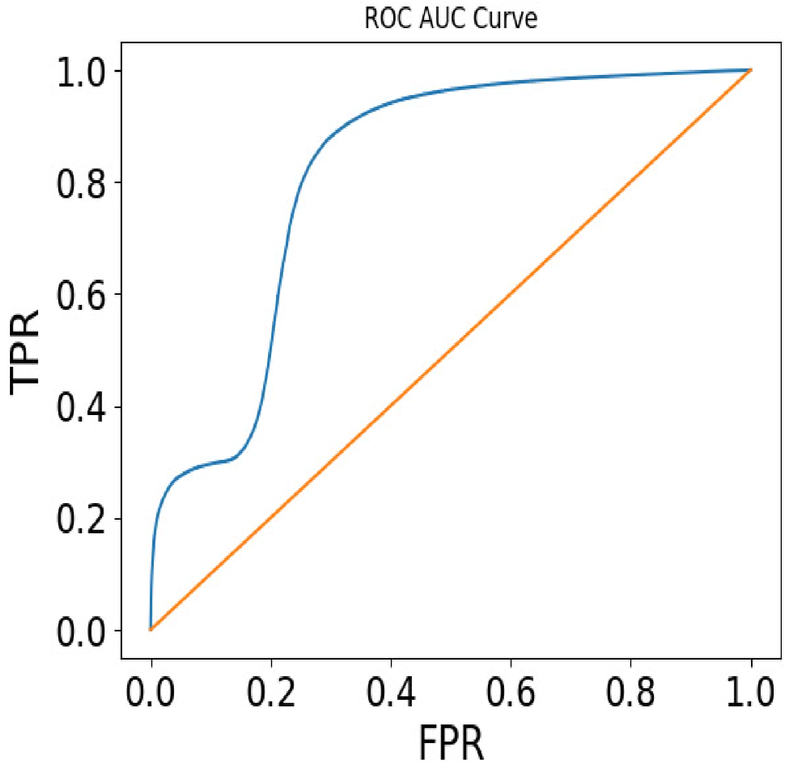


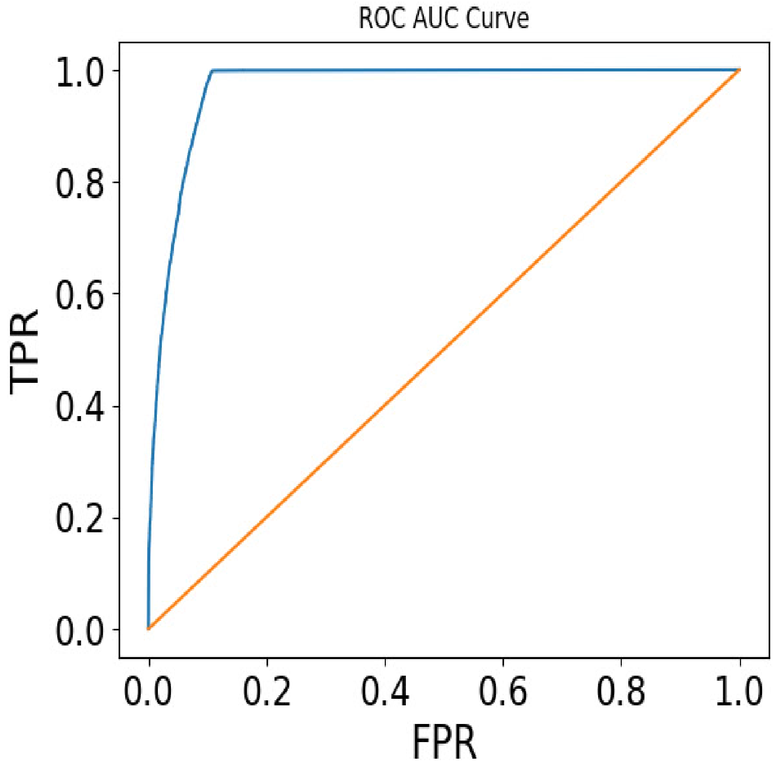
$$TPR = \frac{TP}{TP + FN} \qquad FPR = \frac{FP}{FP + TN}$$



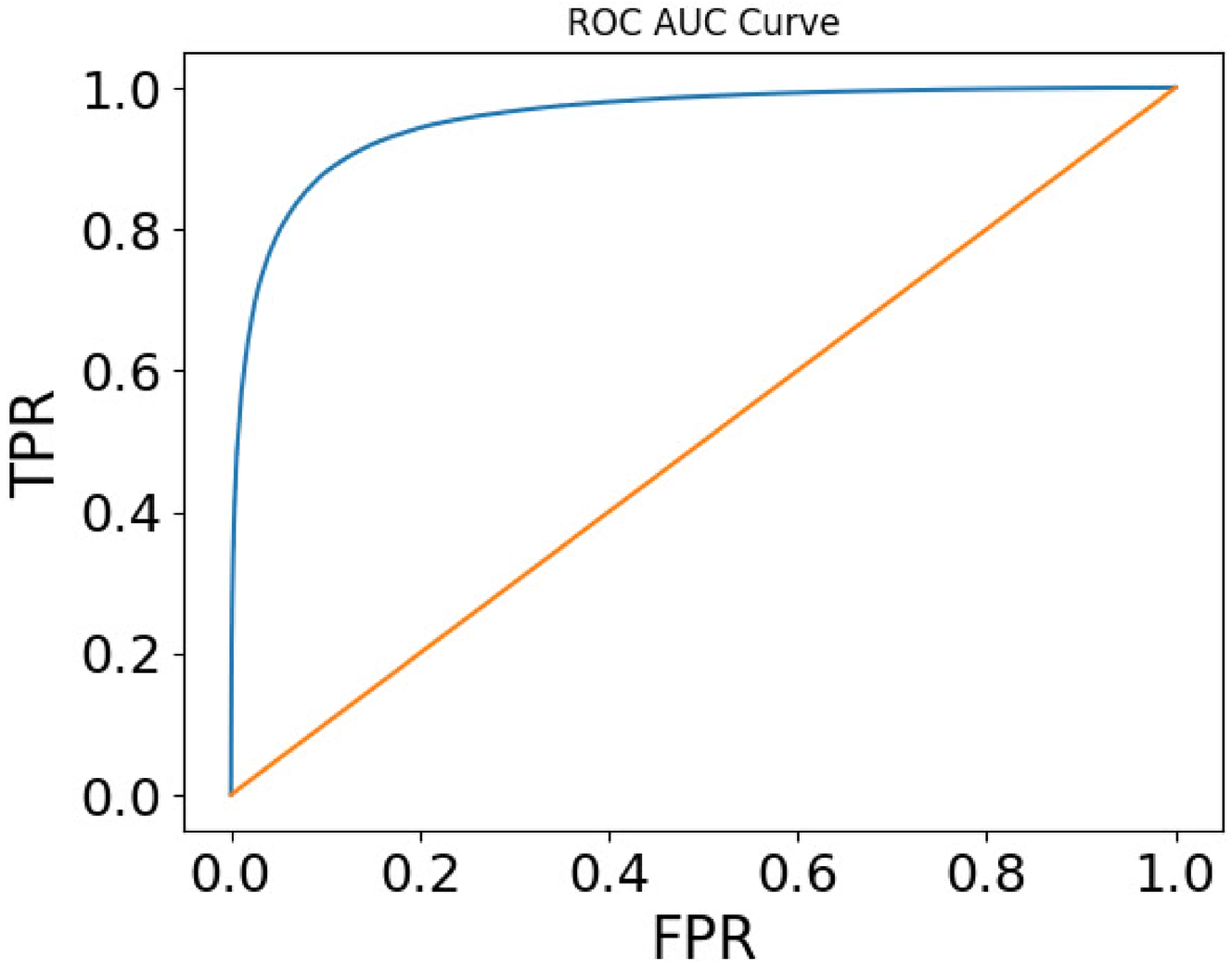
Quality metrics







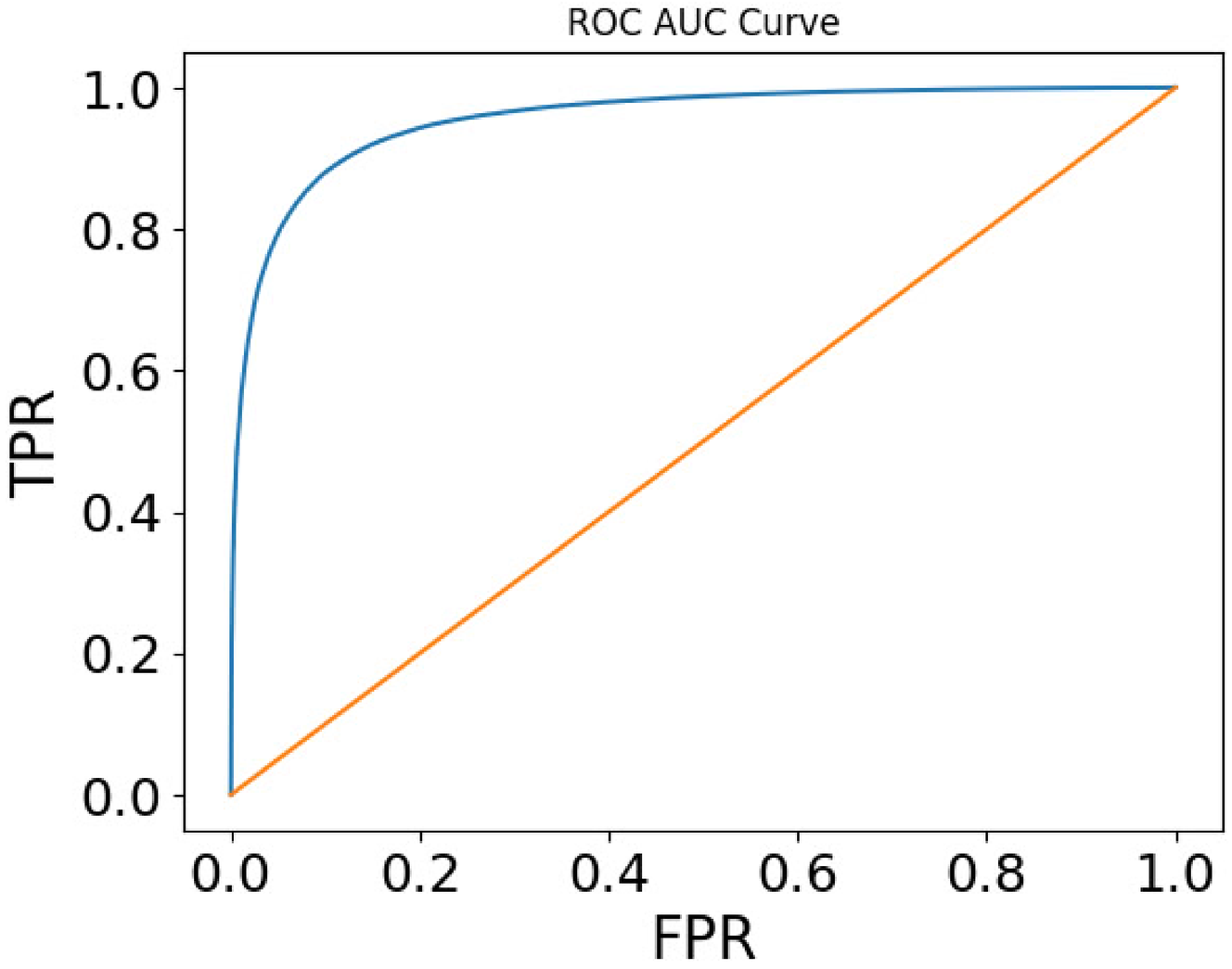
Quality metrics




```
parameters = {
    "loss":["log_loss"],
    "learning_rate": [0.01, 0.2],
    "min_samples_split": [0.01, 0.2],
    "max_depth":[8,12,16],
    "max_features":["log2","sqrt"],
    "criterion": ['squared_error'],
    "n_estimators":[10,20]
    }

clf = GridSearchCV(classif, parameters, cv=3, n_jobs=8)
```

Quality metrics



Choose wisely

1 Specific to the analysis

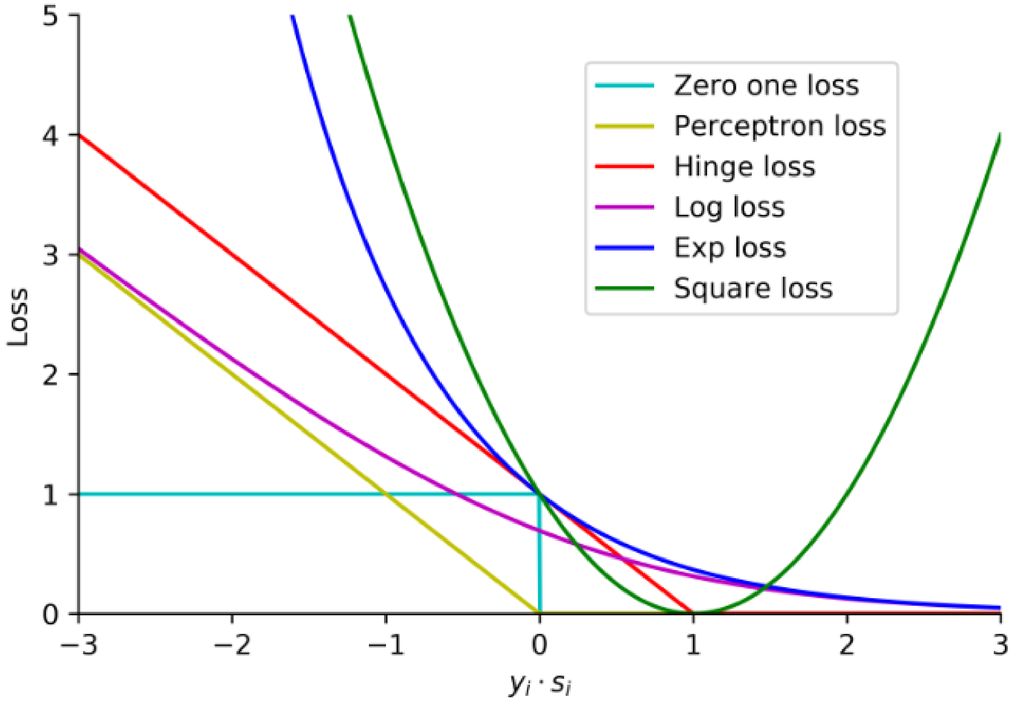
2 Less pars - better

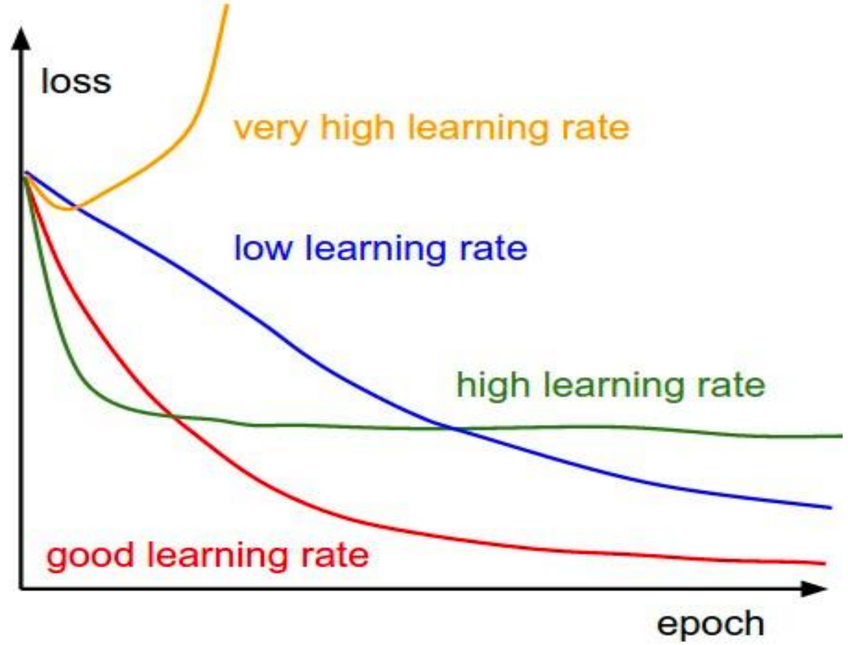
3 Don't let machine choose for you

4 Multiple treshlods – systmetic uncertainties

Regression

Regression





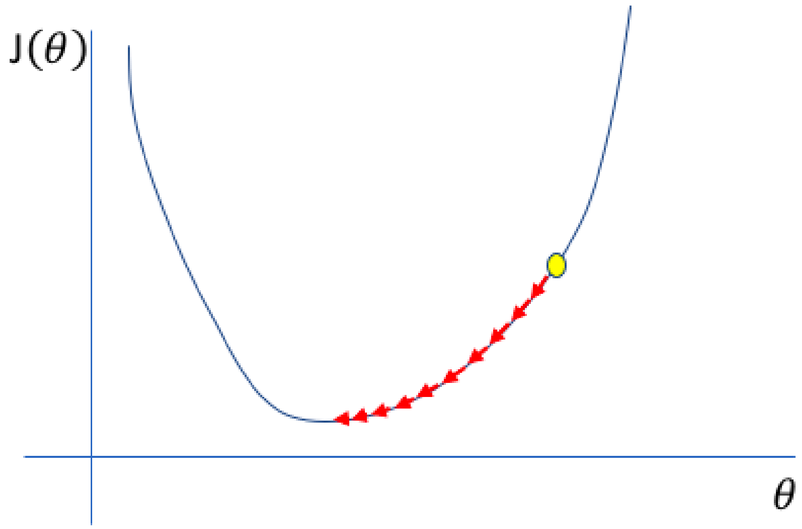
Choose params smartly $\omega_1, \omega_2, .., \omega_n \rightarrow \omega_1^{-1}, \omega_2^2, .., \omega_n$

Choose loss func and solver

Regression

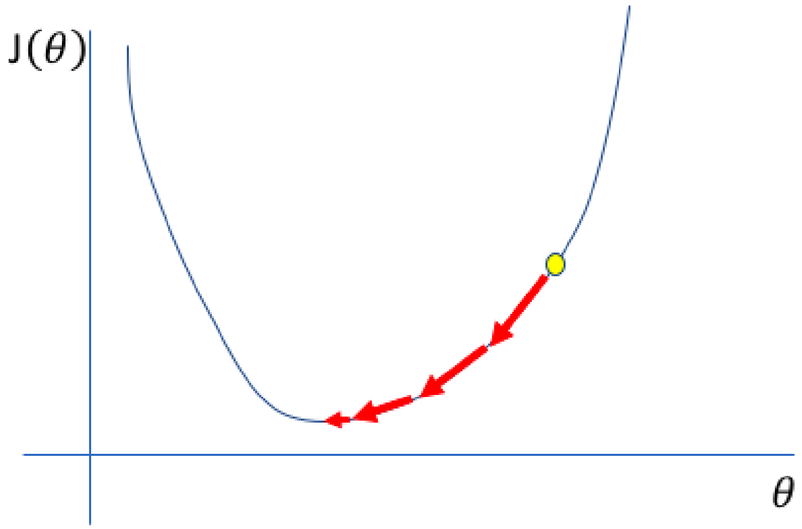
Stony Brook University

Too low



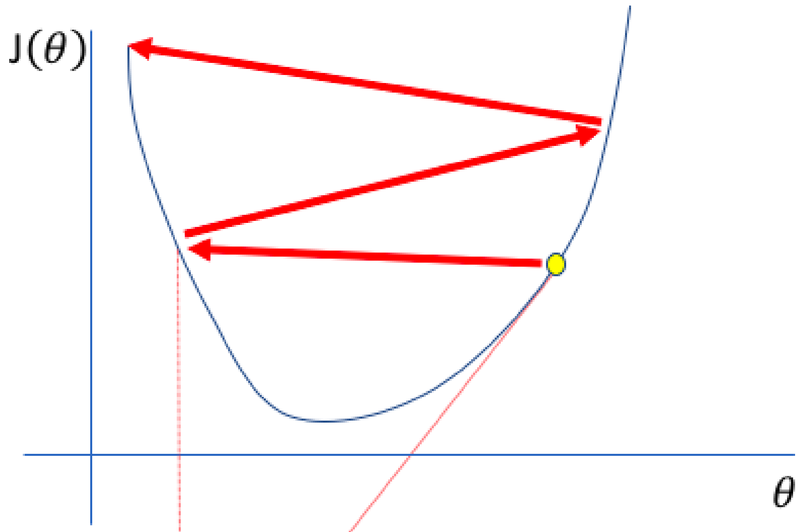
A small learning rate requires many updates before reaching the minimum point

Just right



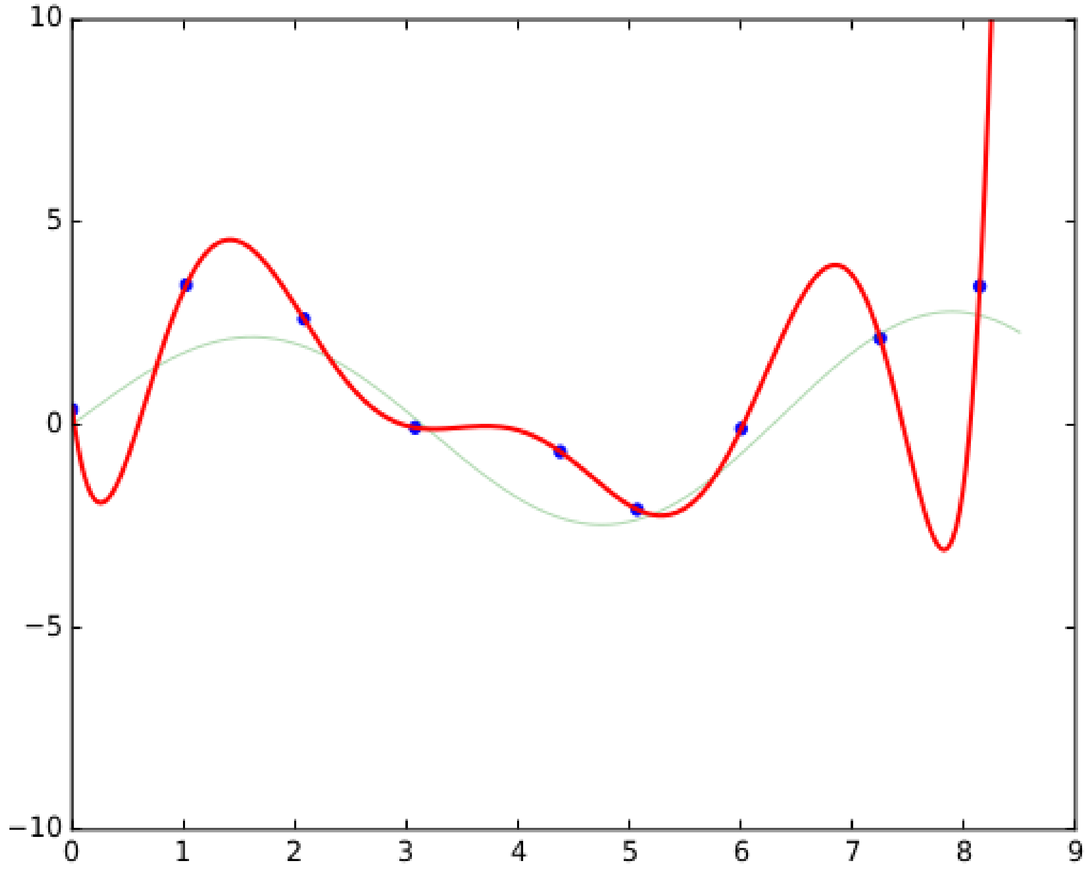
The optimal learning rate swiftly reaches the minimum point

Too high



Too large of a learning rate causes drastic updates which lead to divergent behaviors

Regression



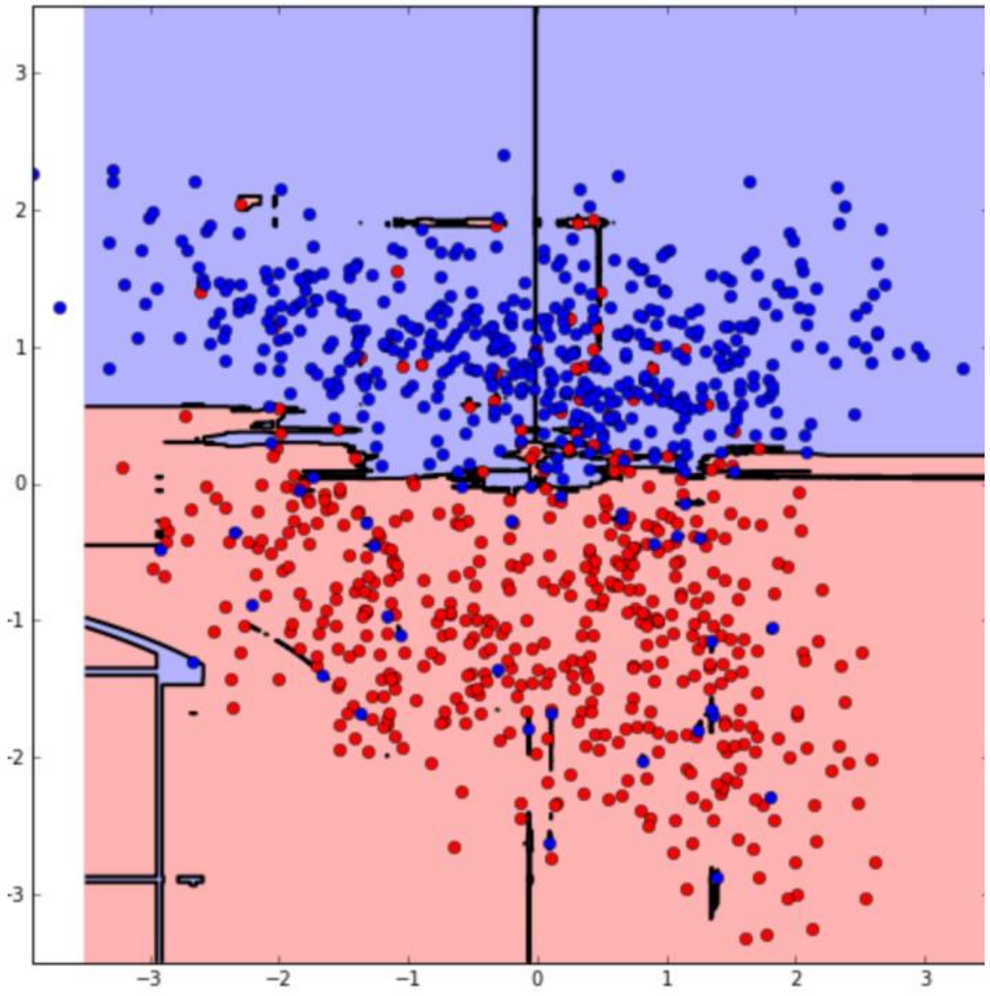
Transforming loss function by adding regularization term

$$\sum_{i=0}^{n}\left(y_i - \omega_0 - \sum_{j=0}^{n}\omega_j\, x_{ij}\right)^2 \rightarrow$$

$$\sum_{i=0}^{m}\left(y_i - \omega_0 - \sum_{j=0}^{n}\omega_j\, x_{ij}\right)^2 + \lambda\sum_{j=0}^{m}|\omega_j|$$

Random Forest

Random Forest



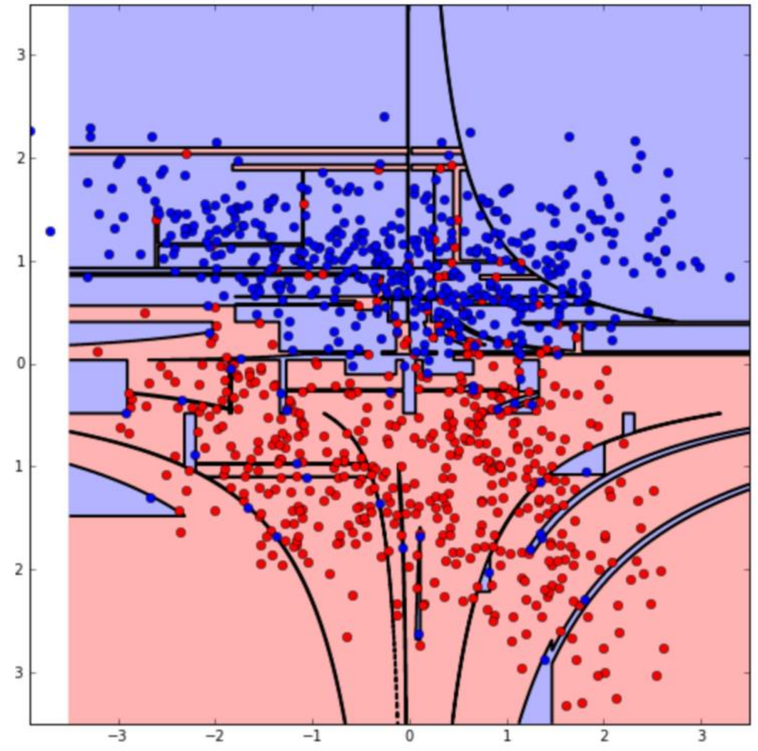
Pros:

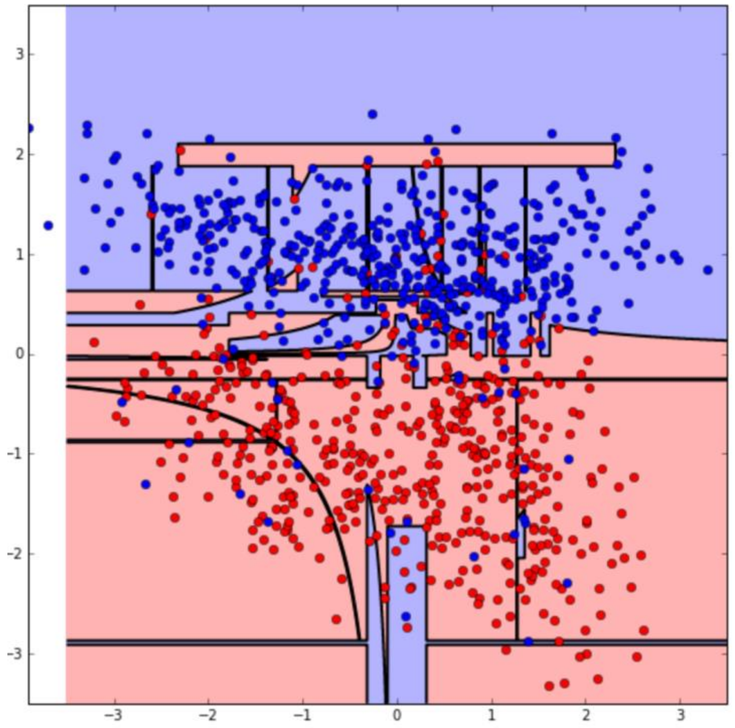
1. Robust to overfitting
2. Easy to use
3. Parallelizable
4. Classification and regression

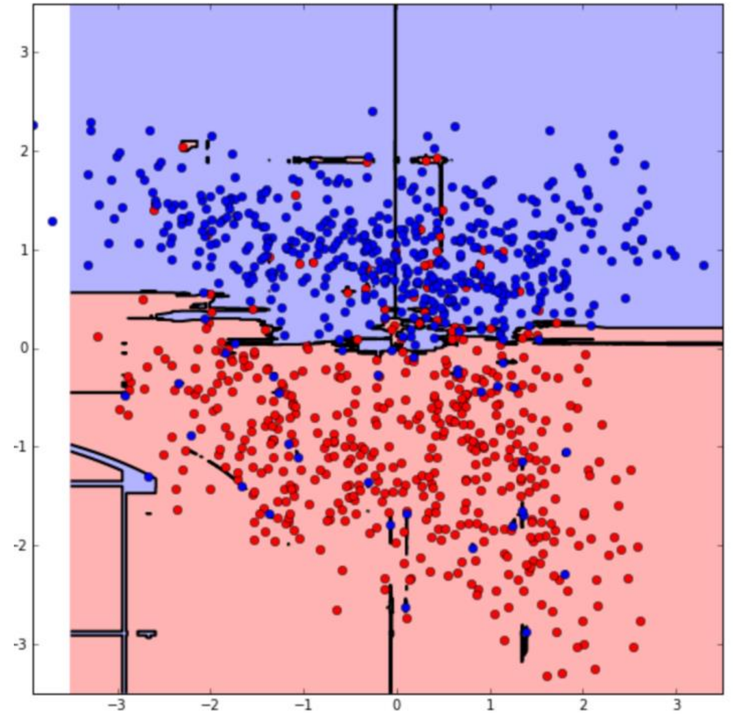
Cons:

1. Harder to interpret
2. Slower and memory intense
3. Less accurate for small dataset

Random Forest







Random Forest

Stony Brook University

n_estimators – Number of trees; increase for better performance but higher computation.

max_depth – Maximum depth of trees; controls overfitting (deeper trees capture more complexity).

min_samples_split – Minimum samples to split a node; higher values prevent overfitting.

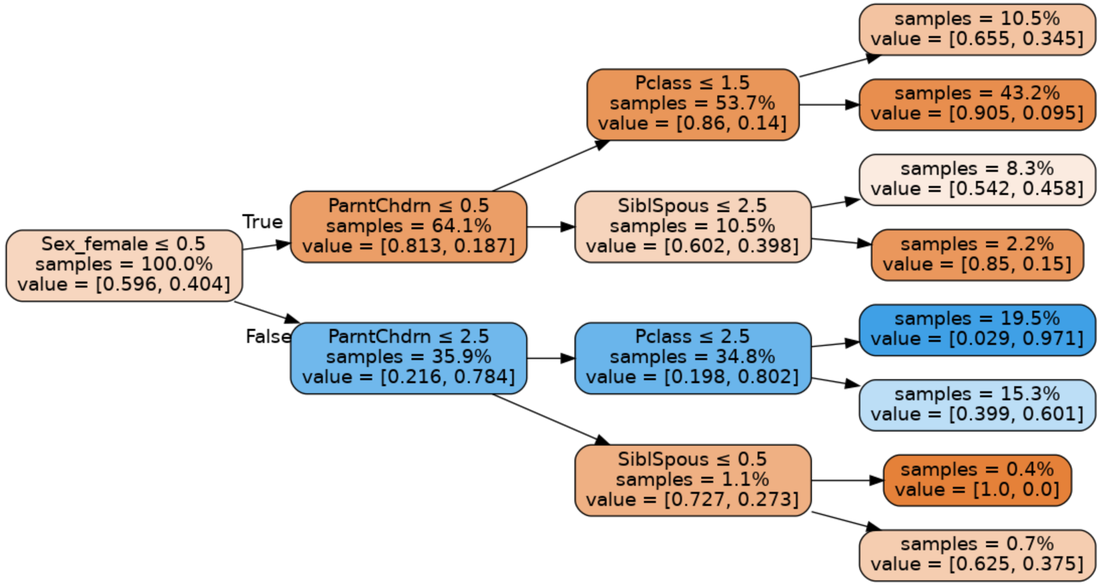
min_samples_leaf – Minimum samples in a leaf; larger values create simpler models.

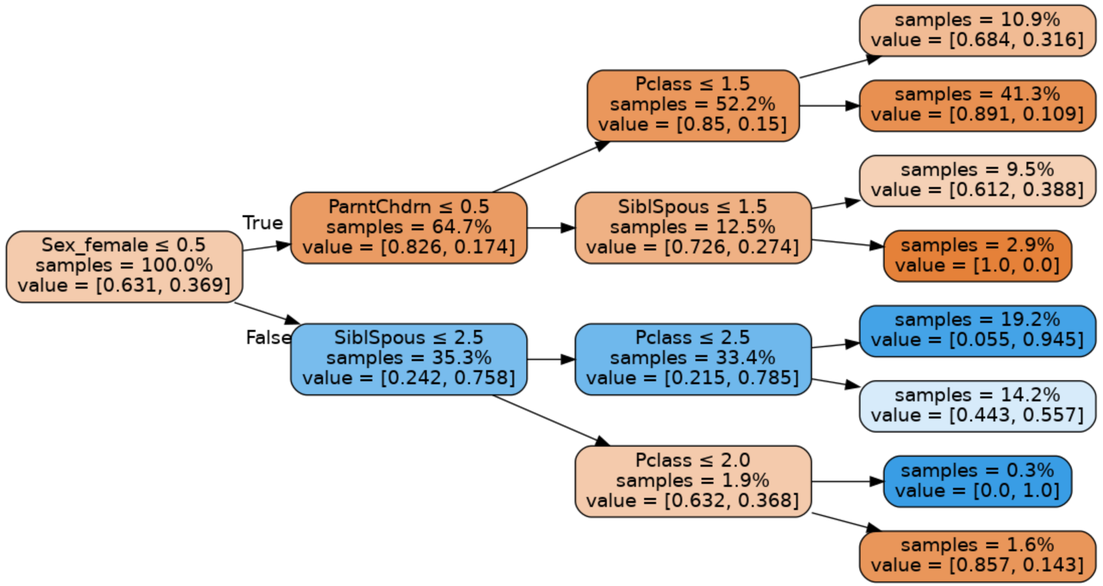
max_features – Controls how many features the model looks at when making each split; choosing fewer features makes the model simpler and faster, while more features make it more accurate but slower.

bootstrap – Whether to sample data with replacement; affects diversity of trees.

criterion – Function to measure quality of a split (e.g., "gini" for classification, "mse" for regression).

Random Forest





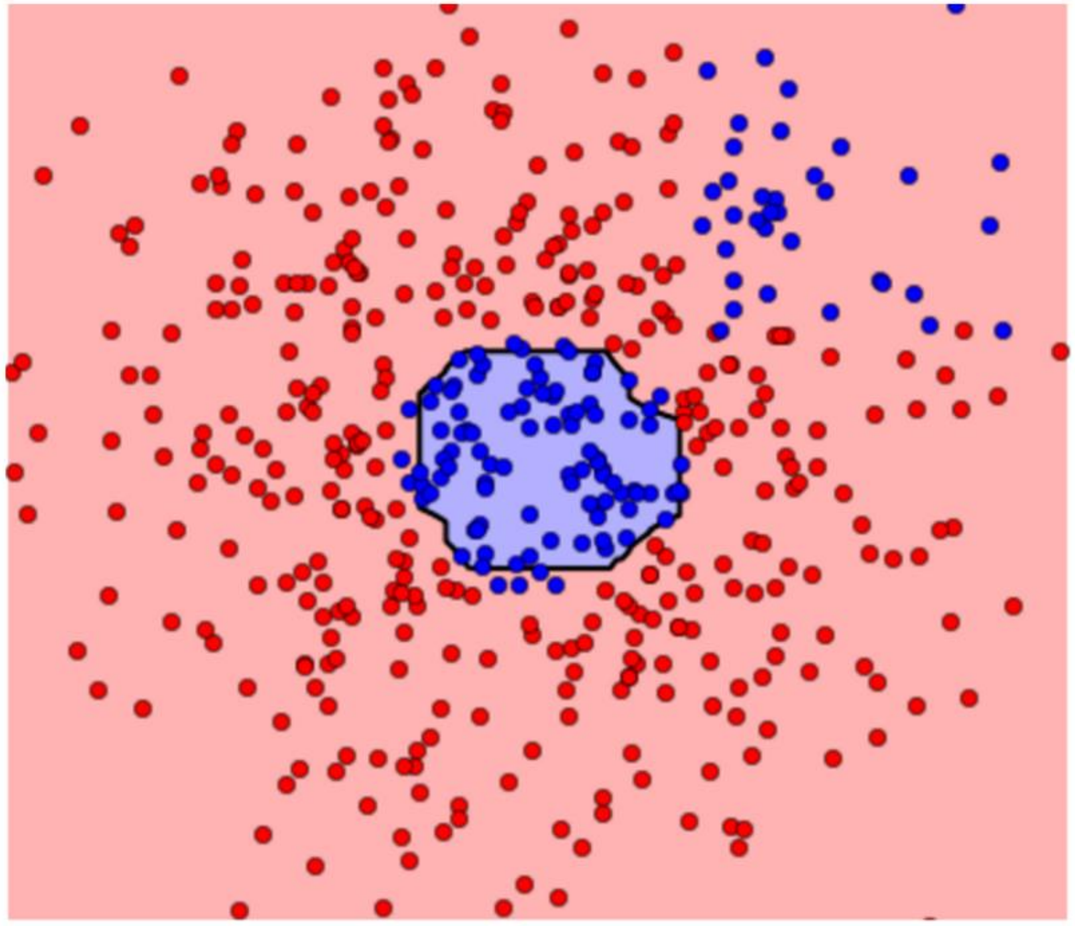
Gradient Boosting

Gradient Boosting

Stony Brook University

Next is fixing

previous issues



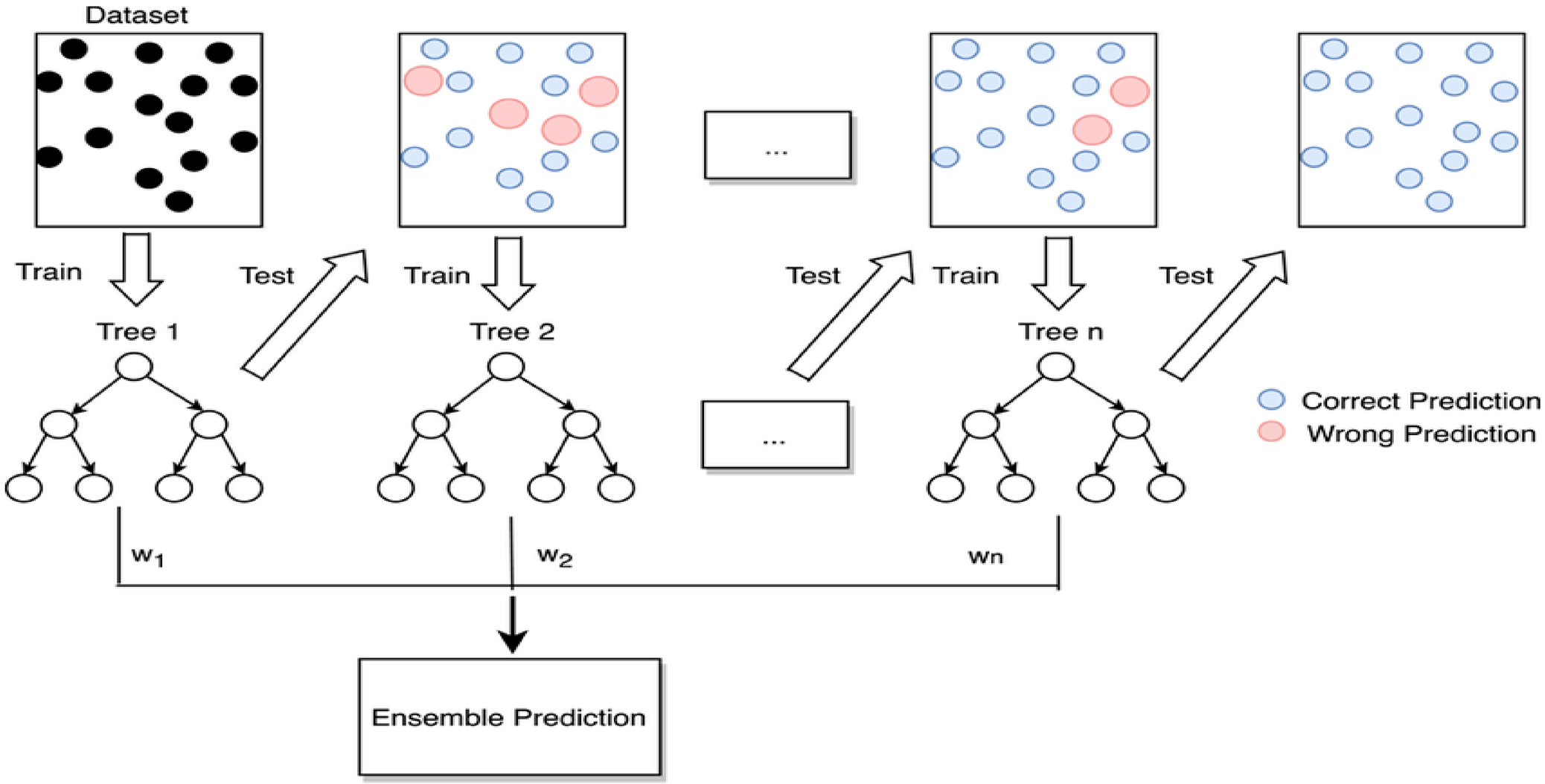
Pros:

1. High accuracy
2. Handle complex tasks
3. Robust to outlier and -9999
4. Classification and regression

Cons:

1. Hard to interpret
2. Slower training
3. Sensitive to overfitting

Gradient Boosting



Gradient Boosting

Stony Brook University

n_estimators – Number of boosting stages; too many can overfit, too few underfit.

learning_rate – Shrinks contribution of each tree; lower values -> more trees → more overfit or precision.

max_depth – Limits tree depth; helps control overfitting.

subsample – Fraction of data used for fitting each tree; reduces variance.

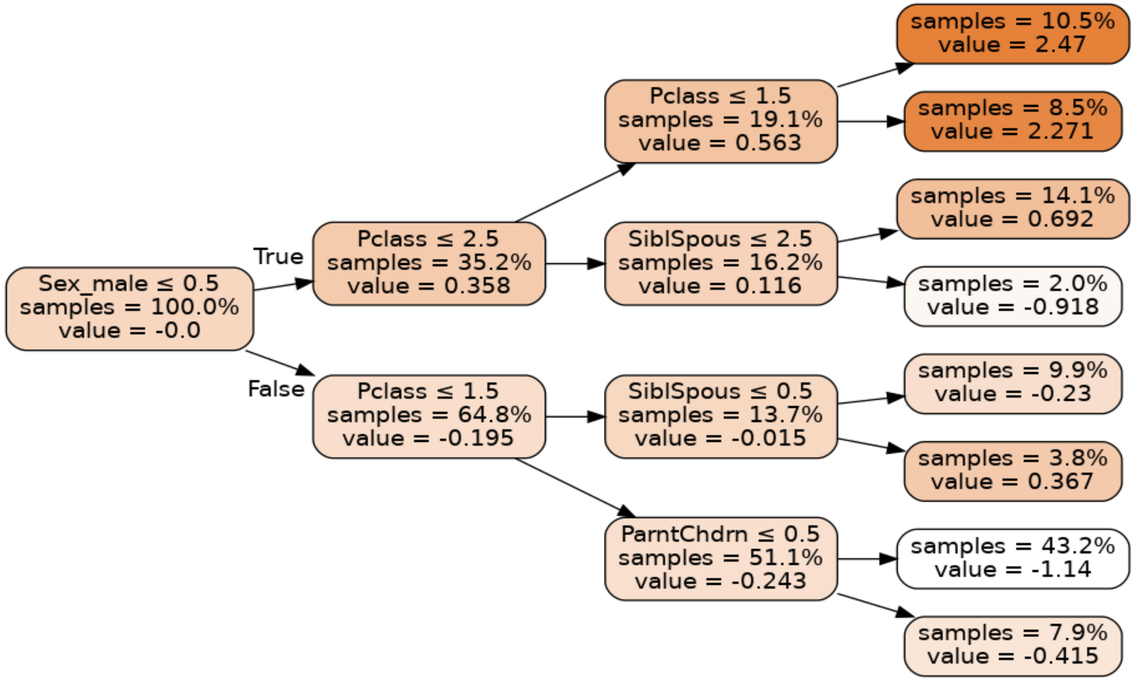
colsample_bytree – Fraction of features used per tree; improves generalization. Or one can use max_features

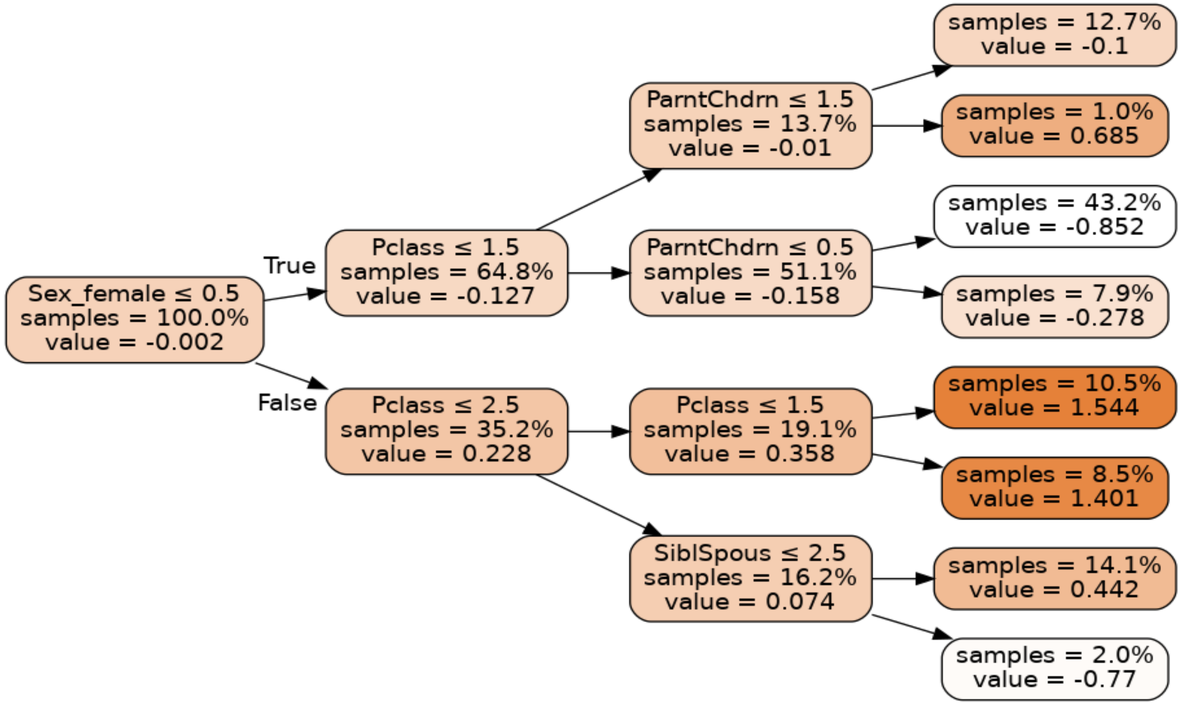
min_samples_split – Minimum samples to split a node; prevents overfitting.

loss – Specifies the loss function to minimize (e.g., "squared_error" for regression, "log_loss" for classification).

criterion – Measures the quality of a split within each tree (e.g., "friedman_mse" for regression).

Gradient Boosting





Outline

Stony Brook University

I asked ChatGPT to summarize ML in HEP:

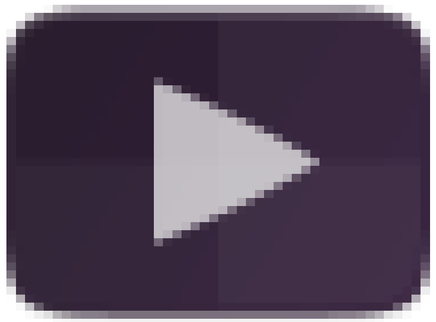
- Classification and Particle Identification
- Data Reconstruction
- Anomaly Detection
- Simulations and Optimization
- Data Reduction and Selection
- Hyperparameter Tuning
- Experimental Designs

Outline

I asked ChatGPT to summarize ML in HEP:

- Classification and Particle Identification
- Data Reconstruction
- Anomaly Detection
- Simulations and Optimization
- Data Reduction and Selection
- Hyperparameter Tuning
- Experimental Designs

Regression



Gradient descent

1