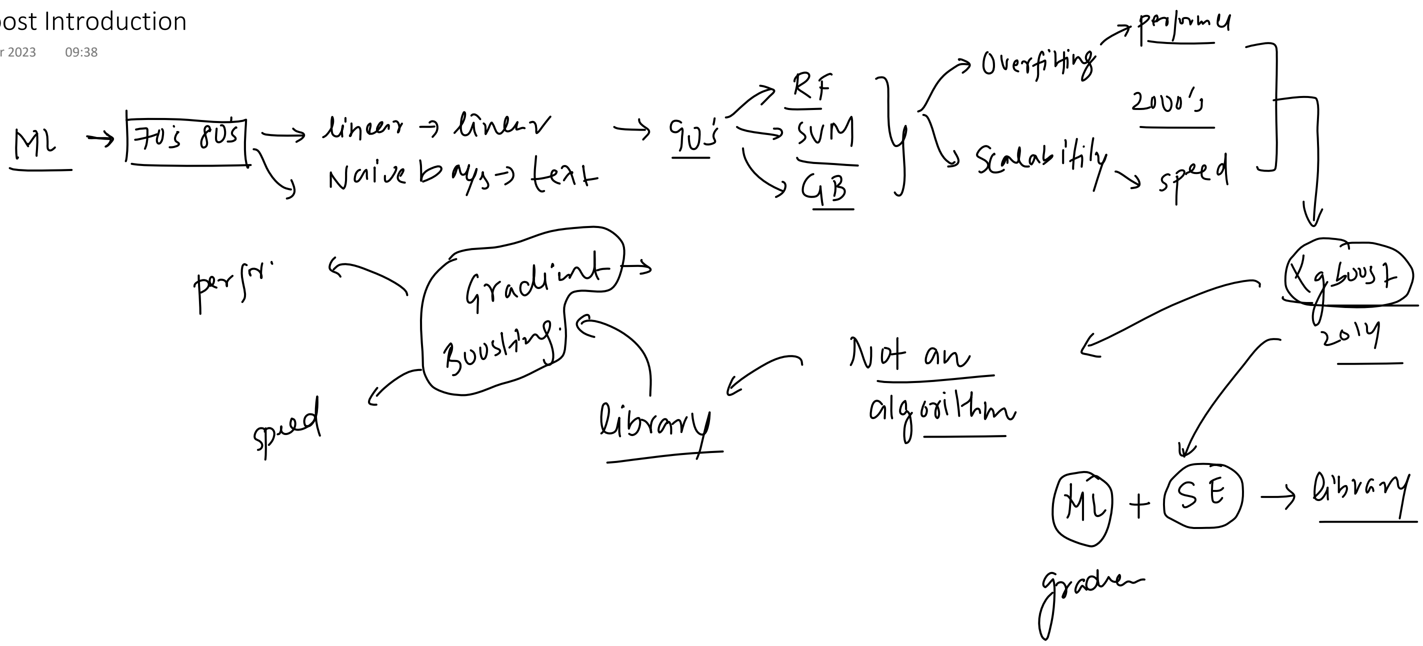
XGBoost Introduction

12 October 2023 09:38



History

12 October 2023 11:01

Early Days → Kaggle pe bawaal → [Open source days] ←

→ (2014)

[Gradient Boostings]

- flexibity (Loss function)
  - reg
  - class
  - ranking
  - custom
- → performance
- → Robust
  - → missing value
  - → Regu
  - perform
  - big data
- → Kaggle → GB

(2016)

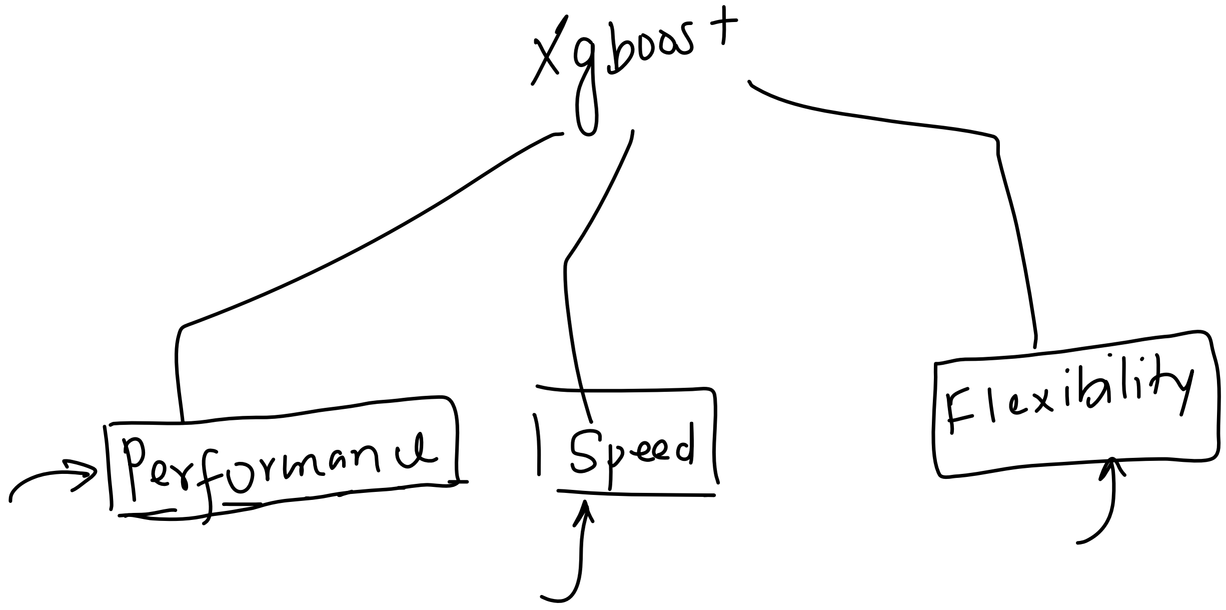
- → Higgs Boson comp (won)
- → 2011 → 29 winning
  - ↓ 16 → Xgboost

2023 → Xgboost
- → perf
- → speed

Tianqi
- feature
- Optimiza
- multiple / prog
- documenta
- Kaggle

XGBoost Features

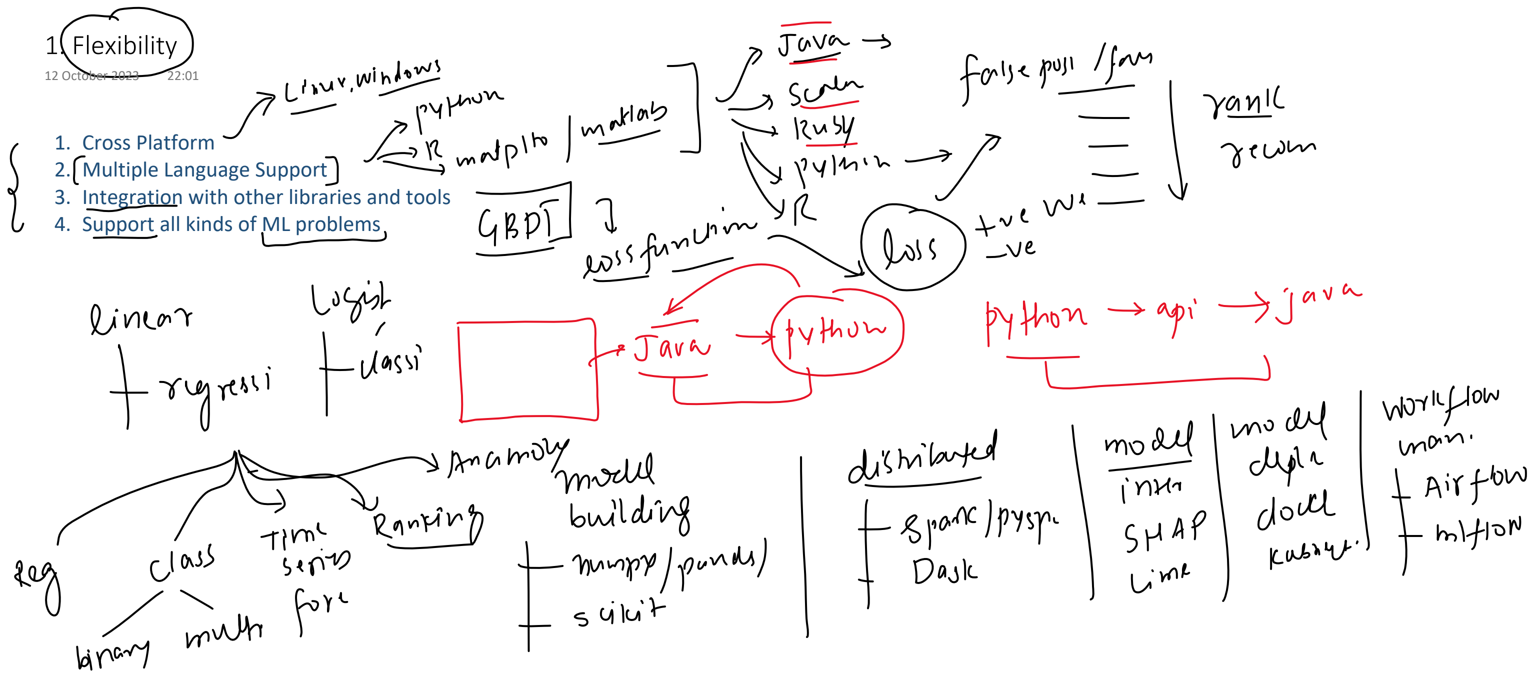
12 October 2023 21:55

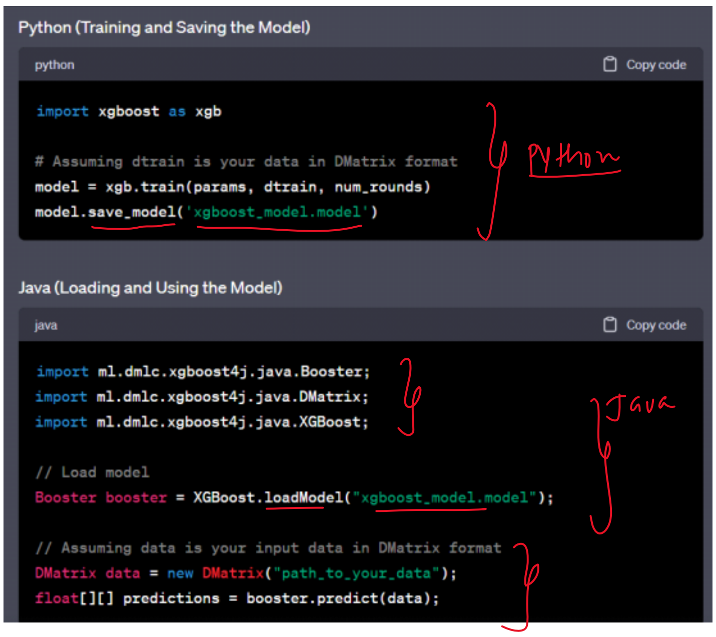


1. Flexibility

12 October 2023 22:01

1. Cross Platform
2. Multiple Language Support
3. Integration with other libraries and tools
4. Support all kinds of ML problems





Python (Training and Saving the Model)

```python
import xgboost as xgb

# Assuming dtrain is your data in DMatrix format
model = xgb.train(params, dtrain, num_rounds)
model.save_model('xgboost_model.model')
```

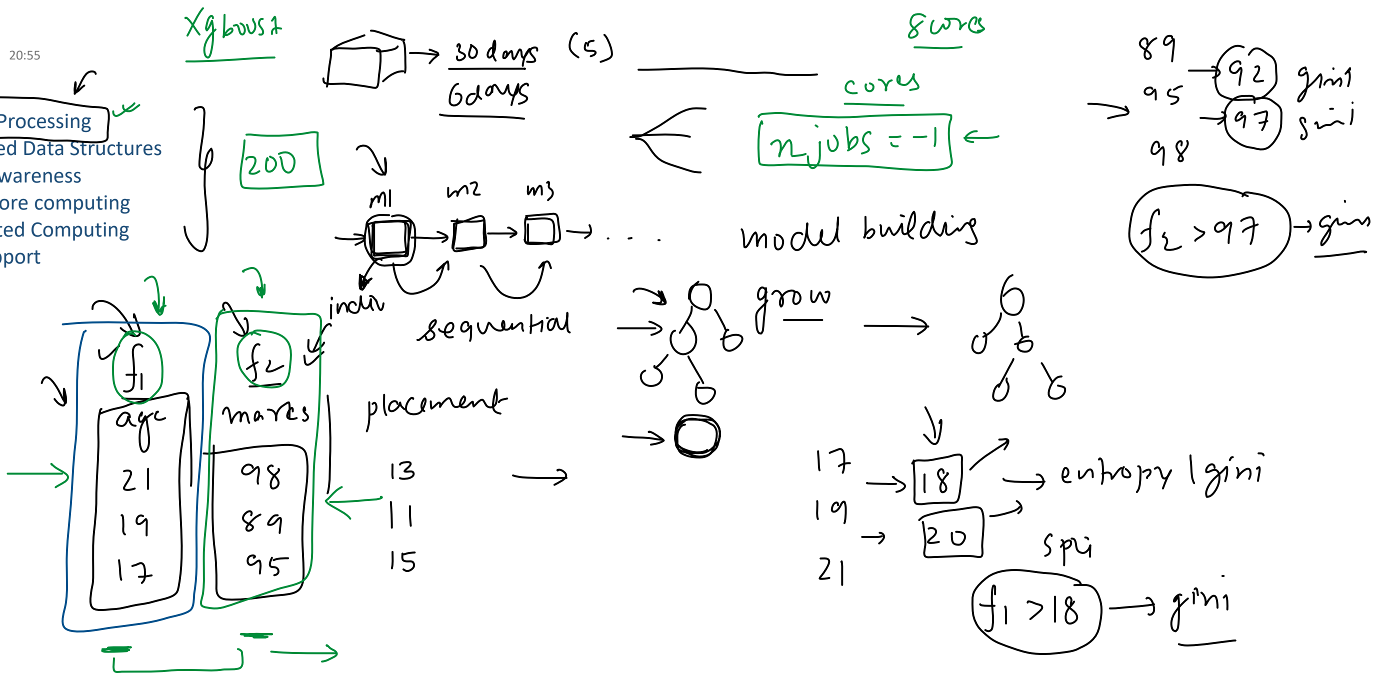
Java (Loading and Using the Model)

```java
import ml.dmlc.xgboost4j.java.Booster;
import ml.dmlc.xgboost4j.java.DMatrix;
import ml.dmlc.xgboost4j.java.XGBoost;

// Load model
Booster booster = XGBoost.loadModel("xgboost_model.model");

// Assuming data is your input data in DMatrix format
DMatrix data = new DMatrix("path_to_your_data");
float[][] predictions = booster.predict(data);
```

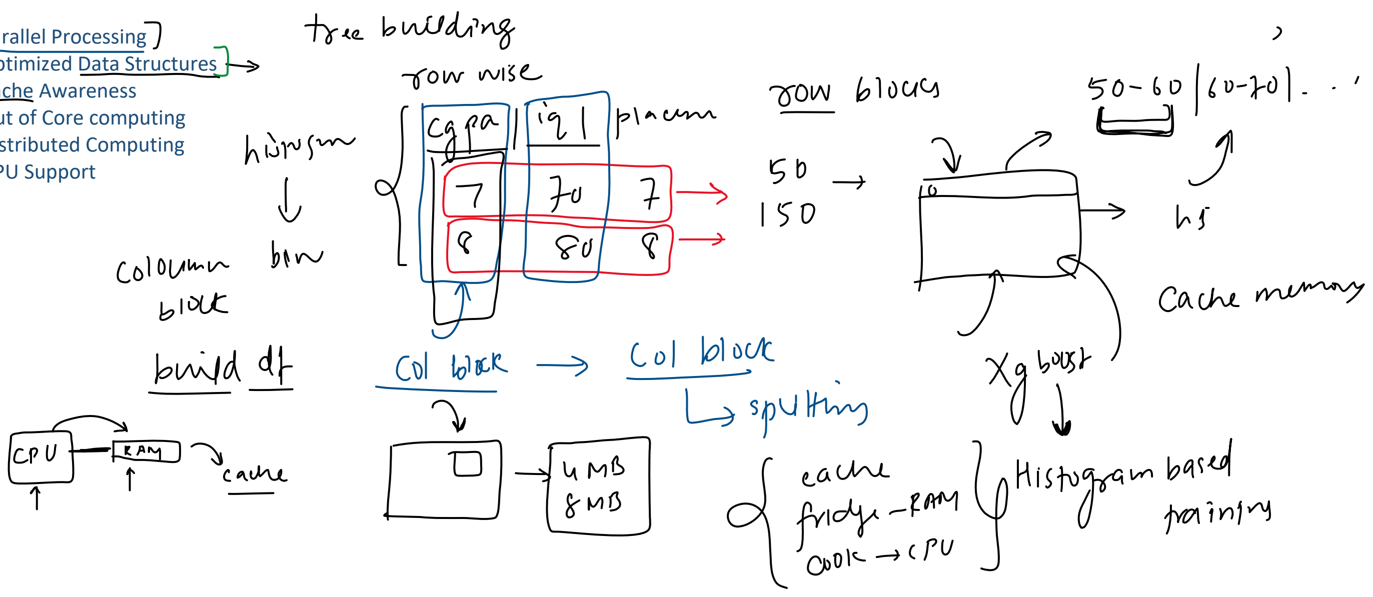
2. Speed

13 October 2023 20:55

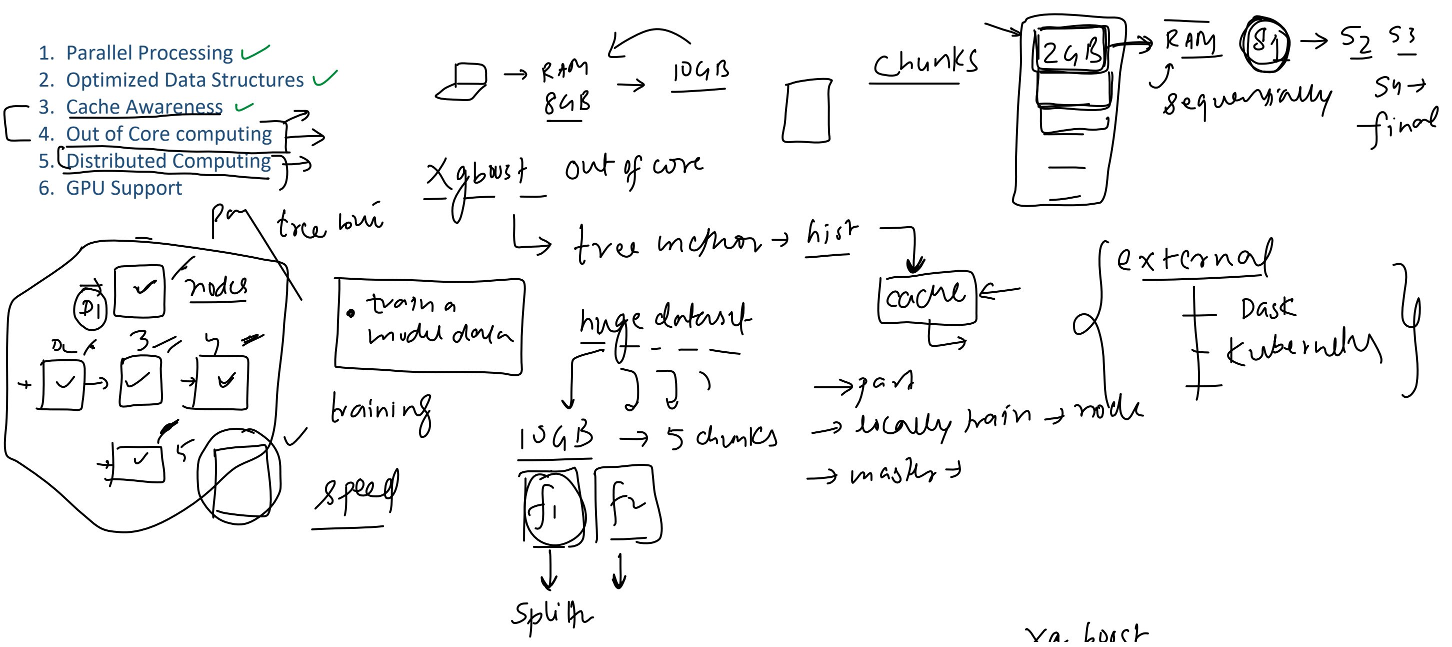
1. Parallel Processing
2. Optimized Data Structures
3. Cache Awareness
4. Out of Core computing
5. Distributed Computing
6. GPU Support



1. Parallel Processing
2. Optimized Data Structures
3. Cache Awareness
4. Out of Core computing
5. Distributed Computing
6. GPU Support



1. Parallel Processing ✓
2. Optimized Data Structures ✓
3. Cache Awareness ✓
4. Out of Core computing
5. Distributed Computing
6. GPU Support



Split

software → Extreme

software

Xg boost
Extreme GB

1. Parallel Processing ✓
2. Optimized Data Structures ✓
3. Cache Awareness ✓
4. Out of Core computing ✓
5. Distributed Computing ✓
→ 6. GPU Support →

CPU → data → CPU → GPU
↳ cores → less powerful
more in number

parallel
histogram split finding

tree_method: gpu_hist

software eng
↓
ML → performance

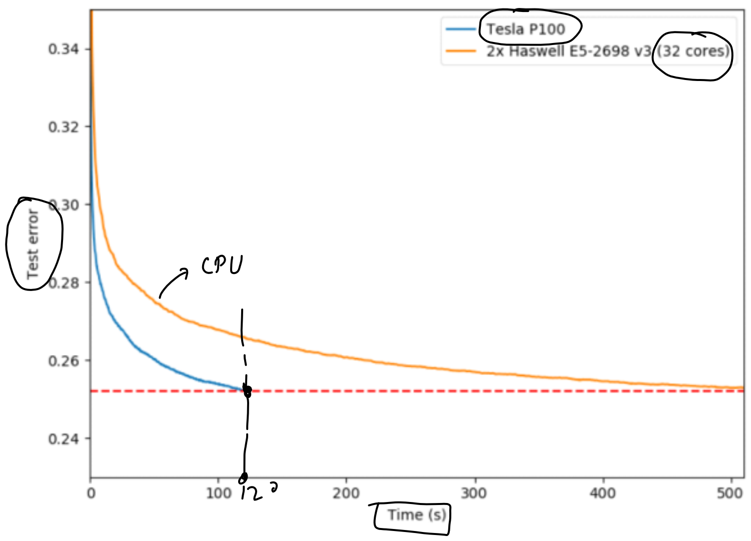


Figure 3. Test error over time for the Higgs dataset, 1000 boosting iterations.

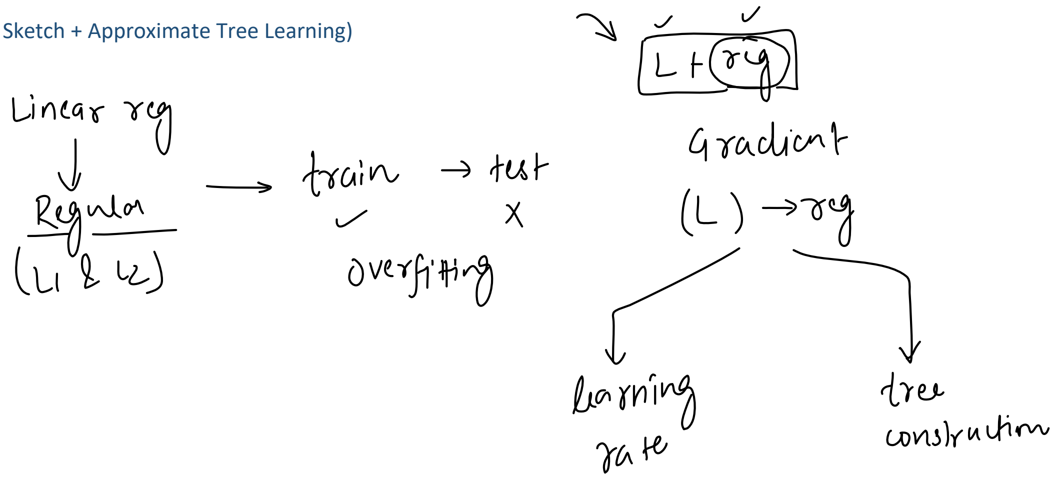
3. Performance →

14 October 2023 10:24

Xgboost

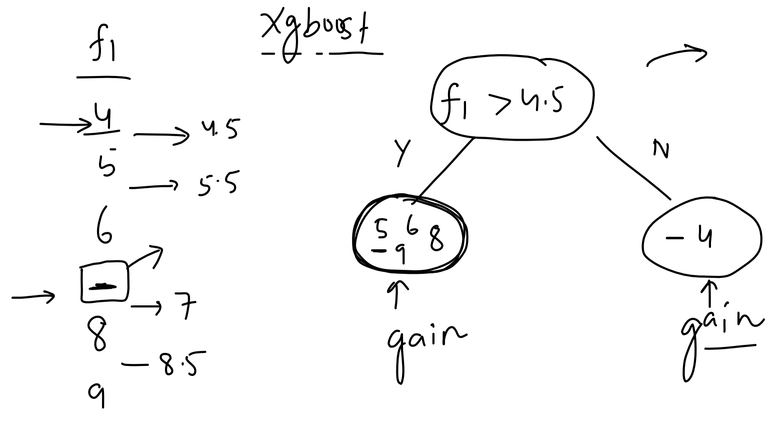
Xgboost → Loss function → min

1. Regularized Learning Objective
2. Handling Missing values
3. Sparsity Aware Split Finding
4. Efficient Split Finding(Weighted Quantile Sketch + Approximate Tree Learning)
5. Tree Pruning



1. Regularized Learning Objective ✓
2. Handling Missing values
3. Sparsity Aware Split Finding
4. Efficient Split Finding(Weighted Quantile Sketch + Approximate Tree Learning)
5. Tree Pruning

ml → missing
↳ prepro
↳ impute



directional sense

1. Regularized Learning Objective ✓
2. Handling Missing values ✓
3. Sparsity Aware Split Finding ✓
4. Efficient Split Finding(Weighted Quantile Sketch + Approximate Tree Learning)
5. Tree Pruning

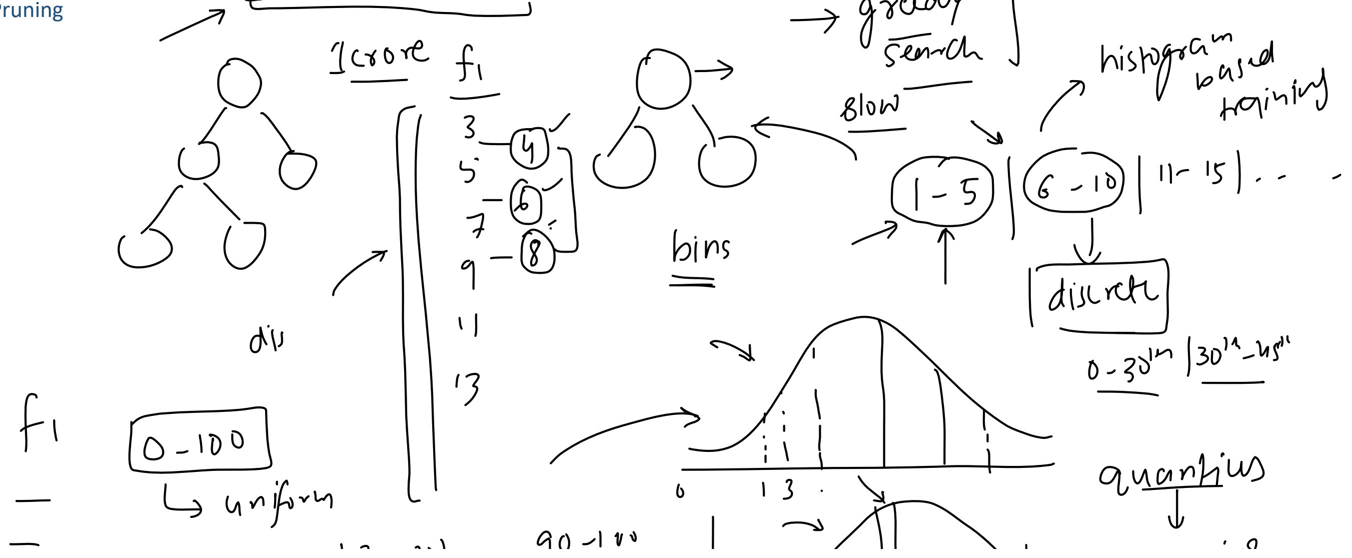
data bins

speed ↑ performance ↓

exact greedy search

slow

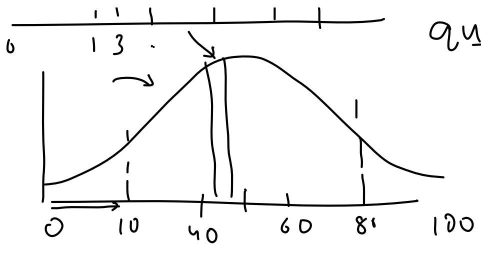
histogram based training



↳ uniform

0-10 | 10-20 | 20-30 | . . 90-100

↳ unif



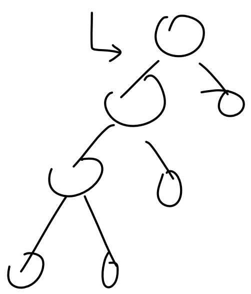
quantiles

↓

binning

GB

1. Regularized Learning Objective ✓
2. Handling Missing values ✓
3. Sparsity Aware Split Finding ✓
4. Efficient Split Finding(Weighted Quantile Sketch + Approximate Tree Learning) ✓
5. Tree Pruning → perform XgBoost



→ reduce complexity

↓

overfitting

→ Post pruning

→ pre pruning

$\boxed{\text{XgBoost}}$

→ prepruning

→ postpr

→ $\boxed{\text{gamma value}}$