AMR optimization

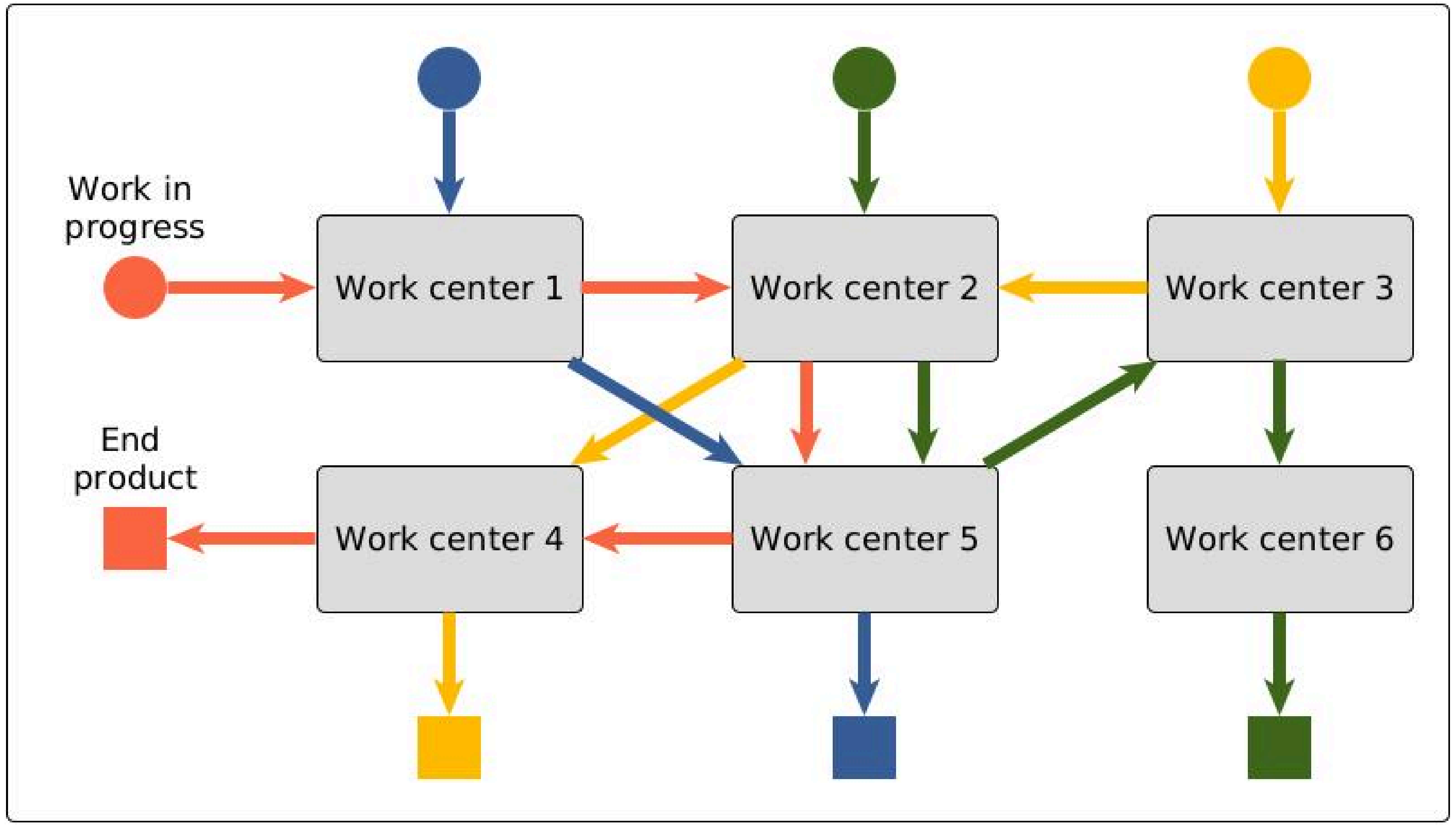
1 Team

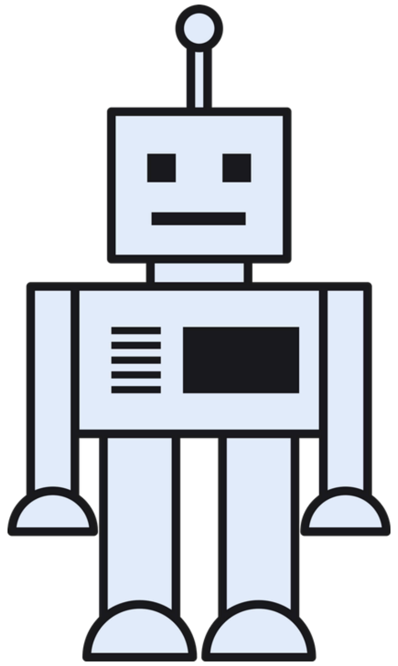
KWON SEOKGEUN
KIM KAEUL
PARK MINJU
JANG JUNHYEOK

Contents

Introduction : motivation of this project

Job shop scheduling

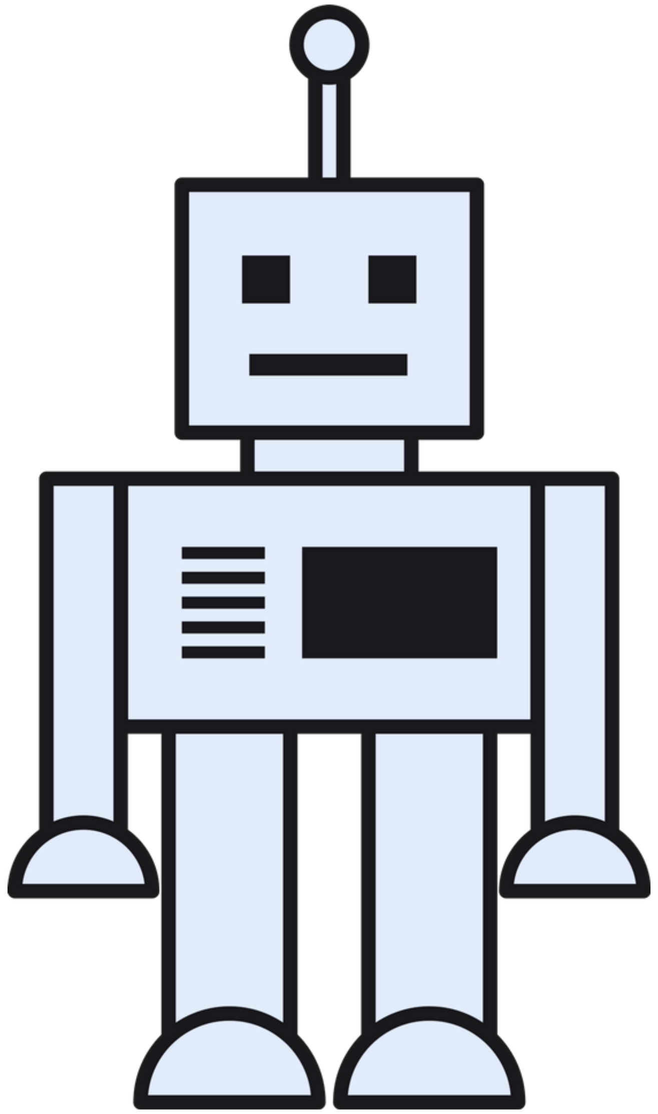




AMR

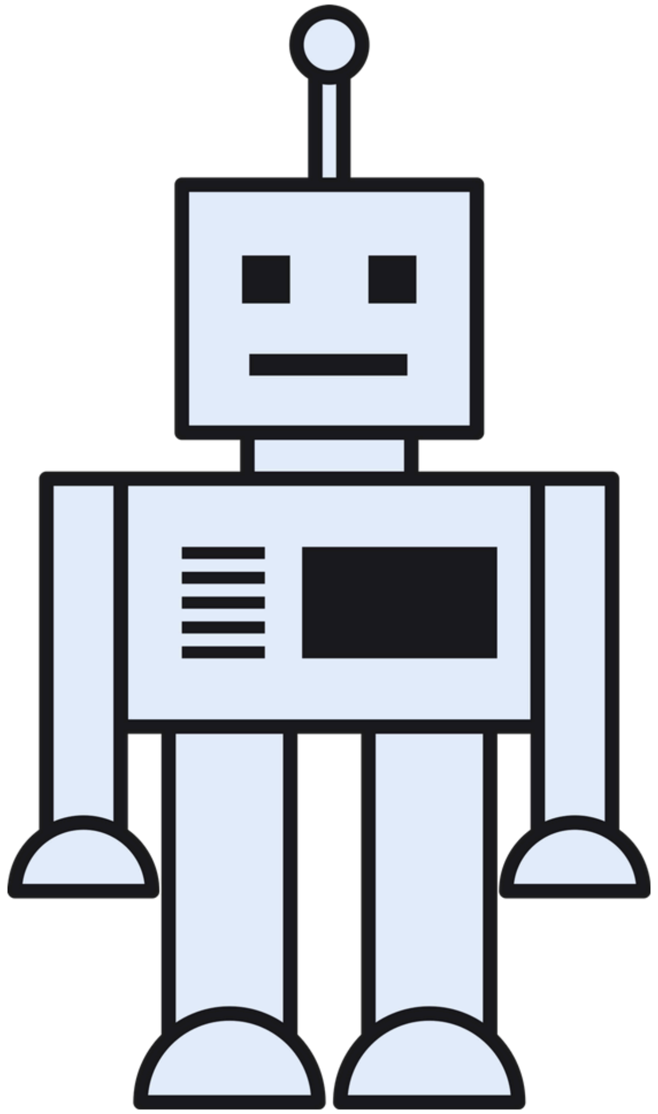
Optimizing or meeting performance criteria

Importance of AMRs in Modern Industry



Importance

- Traditional material handling and transportation methods are often labor-intensive and prone to human error
- Increased efficiency, reduced operating costs and improved accuracy
- Operates 24 hours a day without fatigue

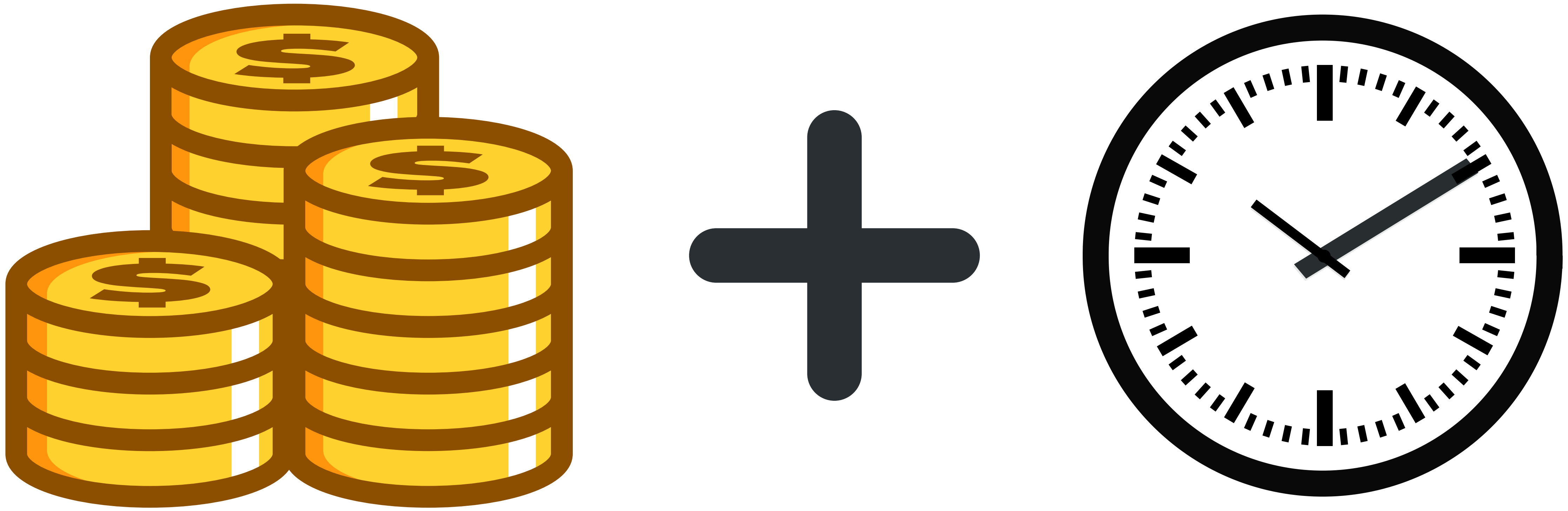


Necessity

- Determine the optimal number of AMRs needed to balance cost and performance
- Choosing the appropriate model and configuration is also important to minimize total tardiness and operational costs
- Explore various optimization techniques to determine the optimal strategy for AMR deployment

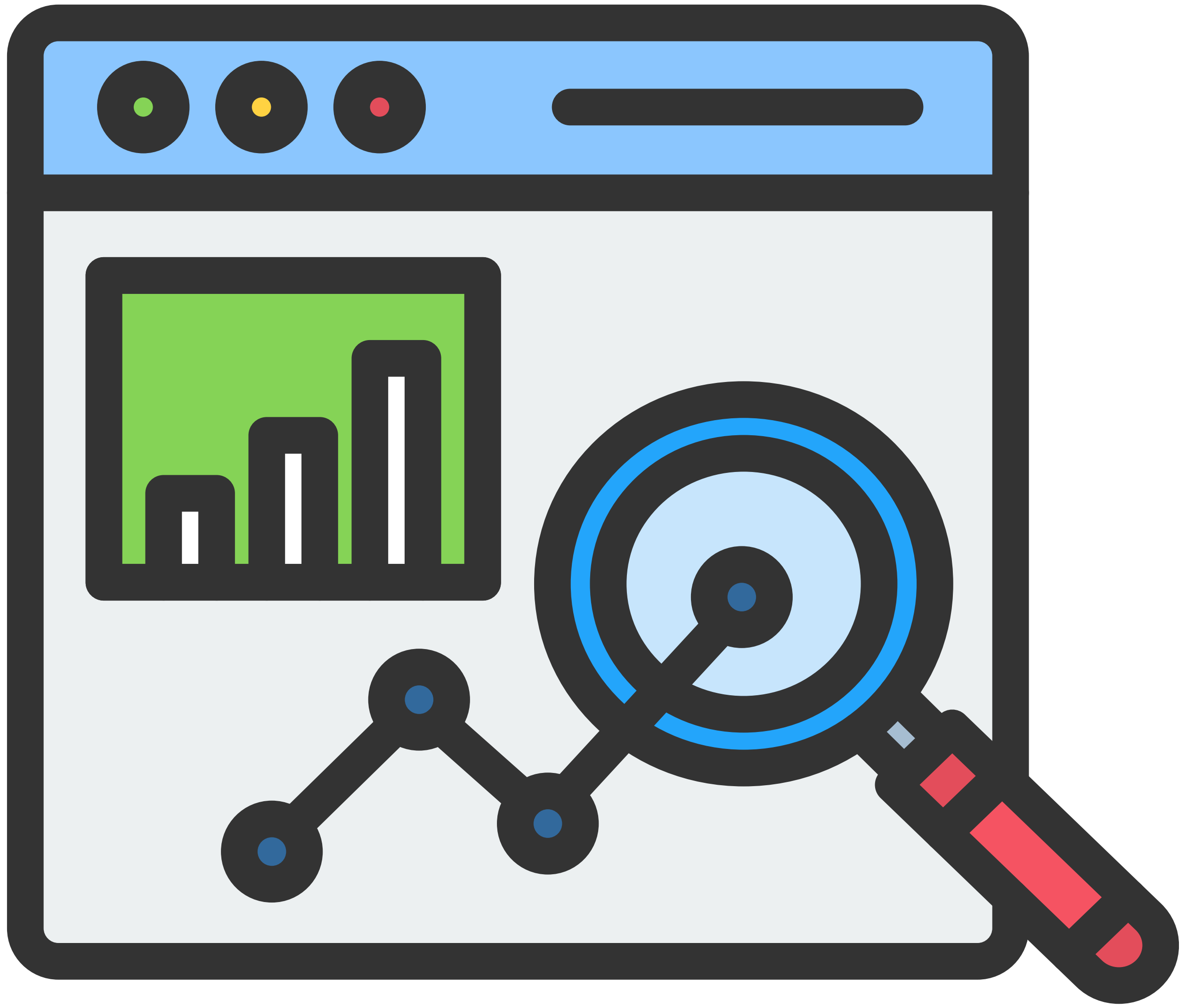
Problem description

Problem description



Optimize time and cost

Problem description



Lowering TotalTardiness by analyzing the number of AMRs
and finding the optimal movement route

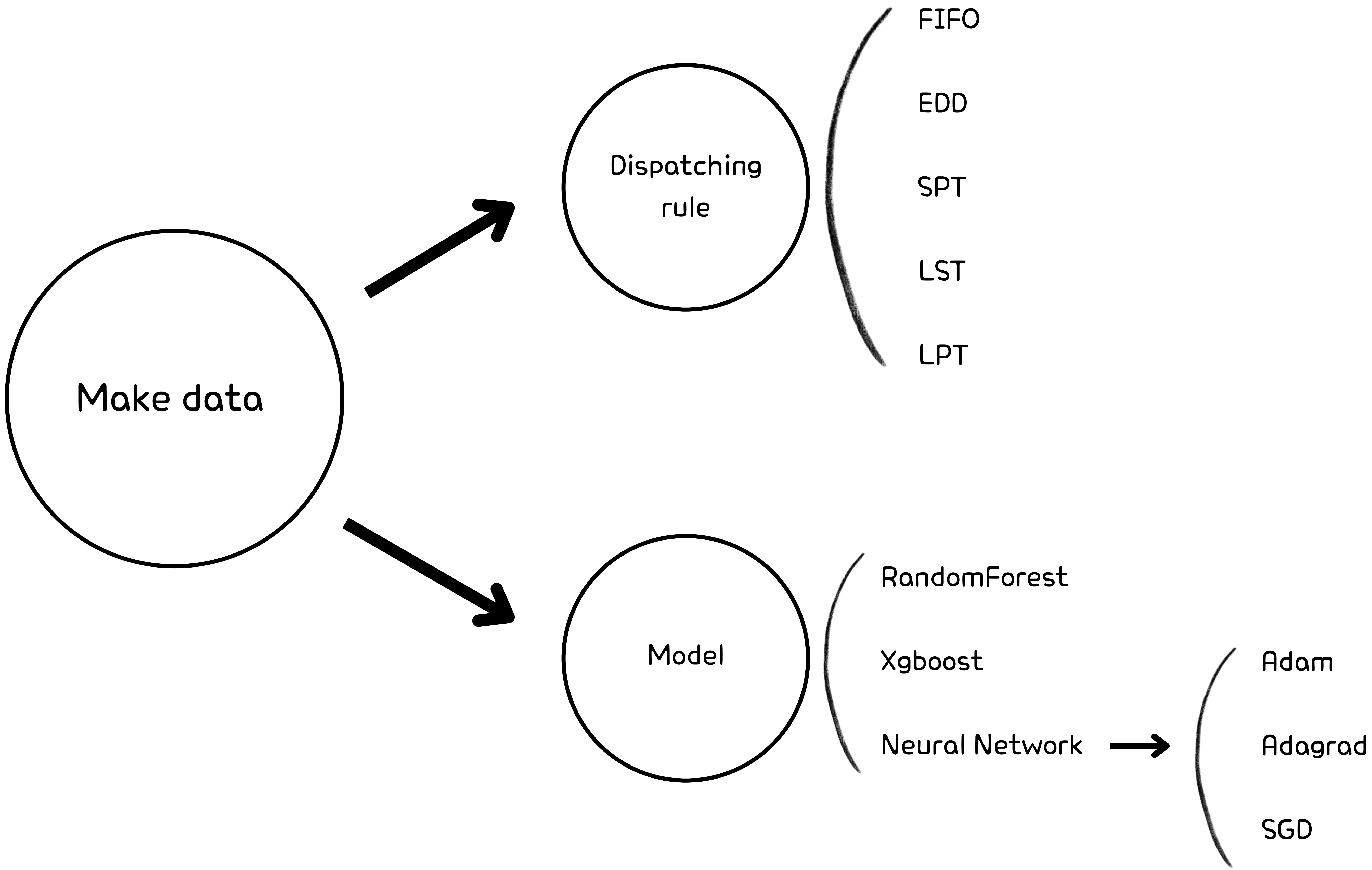
Problem assumption

<u>Six assumptions were made</u>

- All work is prepared from the beginning
- Processing time is fixed
- AMR only handle one task at a time
- Priority constraints between tasks
- Once a task begins, it continues without interruption until completion
- Machine breakdown or maintenance is not taken into account

Methodology

Minimize total tardiness



LST(Least Slack Time)

- LST Rules

: Prioritize the least amount of slack time to work

: Focus on meeting job deadlines

- What's a slack time?

:Time Remaining Before Job Deadline – Processing Time Required to Complete the Job

$$\mathrm{ST}_i = (D_i - t) - P_i$$

- D_i is the due date of job i.
- t is the current time.
- P_i is the processing time of job i.

DispatchingRule

LST(Least Slack Time)

⟨example⟩

A: Due Date = 10, Current Time = 2, Remaining Processing Time = 3

$$\text{Slack Time} = 10 - 2 - 3 = 5$$

B: Due Date = 8, Current Time = 2, Remaining Processing Time = 2

$$\text{Slack Time} = 8 - 2 - 2 = 4$$

⟨code⟩

```
    slackTime = (material.dueDate - currentTime) - material.processTime
if slackTime < leastSlackTime:
        leastSlackTime = slackTime
        leastSlackMaterialIndex = index
```

LPT(Longest Processing Time)

- LPT Rules

：Prioritize tasks with the longest processing time

：Processing longer tasks first ➜ Optimizing the processing time of the system

$$\arg\max_i P_i$$

where:

- P_i is the processing time of job i.

DispatchingRule

LPT(Longest Processing Time)

<example>

A: Processing Time = 5

B: Processing Time = 3

C: Processing Time = 7

<code>

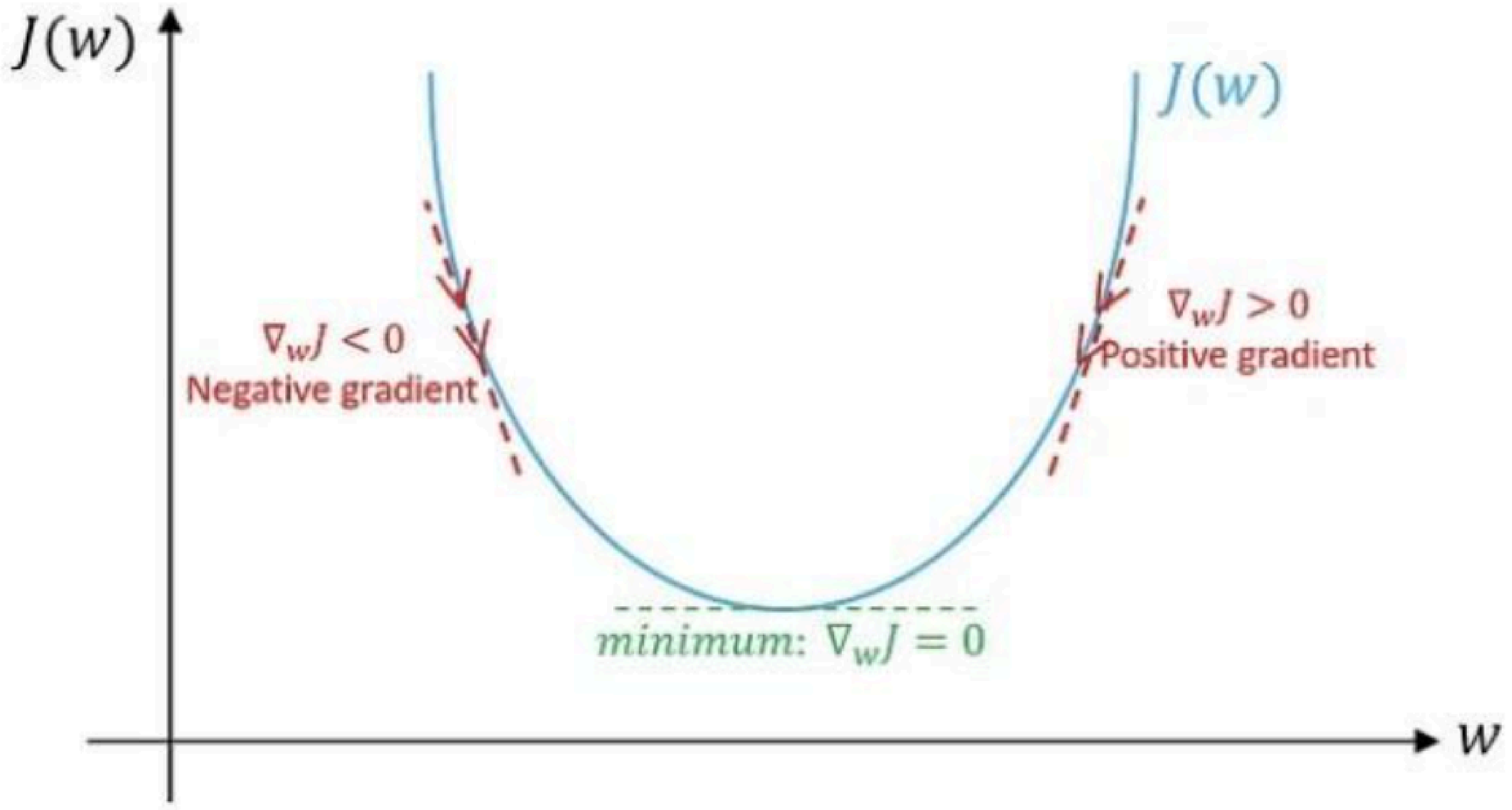
```
if material.processTime > longestProcessTime:
    longestProcessTime = material.processTime
    longestMaterialIndex = index
```

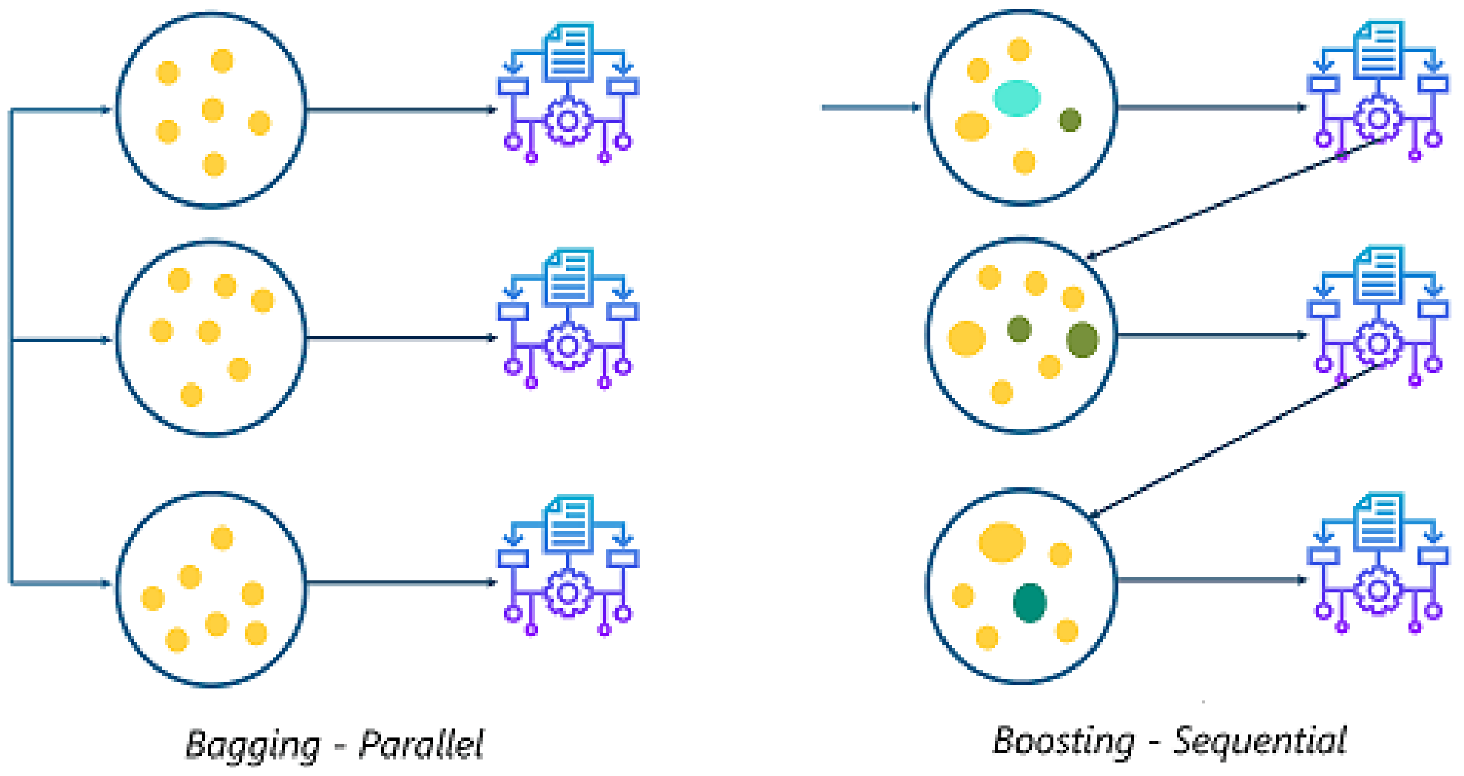
Model

xgboost

- XGBoost（Extreme Gradient Boost）
：Representative algorithms using boosting techniques
：Ensemble techniques that combine vulnerable decision trees

- First, we need to know Gradient Boosting





Model

xgboost

- Library to support parallel learning : XGBoost

: Support both regression and Bunchu problems

: performance, efficiency ↑

- Xgboost execution process

1. Weighing the learning errors of weak models
2. Reflects sequentially to the model
3. Generating a strong predictive model

Model

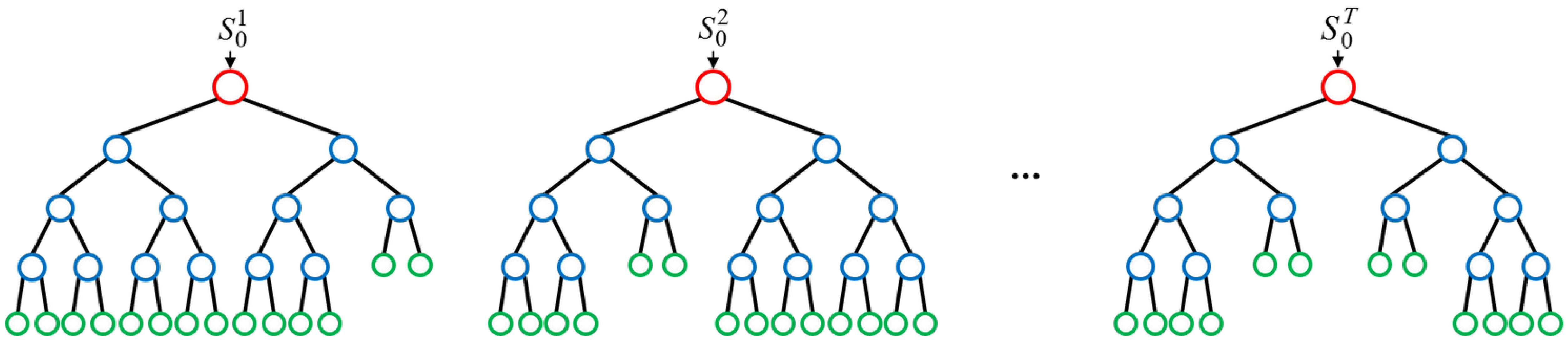
xgboost

<code>

```
from xgboost import XGBClassifier
import xgboost

dtrain = xgboost.DMatrix(data=X_train, label = y_train)
dtest = xgboost.DMatrix(data=X_test, label=y_test)

params = {'max_depth' : 3,
         'eta' : 0.1,
         'eval_metric' : 'logloss',
         'early_stoppings' : 100 }

num_rounds = 400
wlist = [(dtrain, 'train'), (dtest,'eval')]
xgb= xgboost.train(params = params, dtrain=dtrain, num_boost_round=num_rounds, evals=wlist)
```

RandomForest

- Combine basic classifiers (trees) into one classifier (random forest)
  : using average or majority voting method

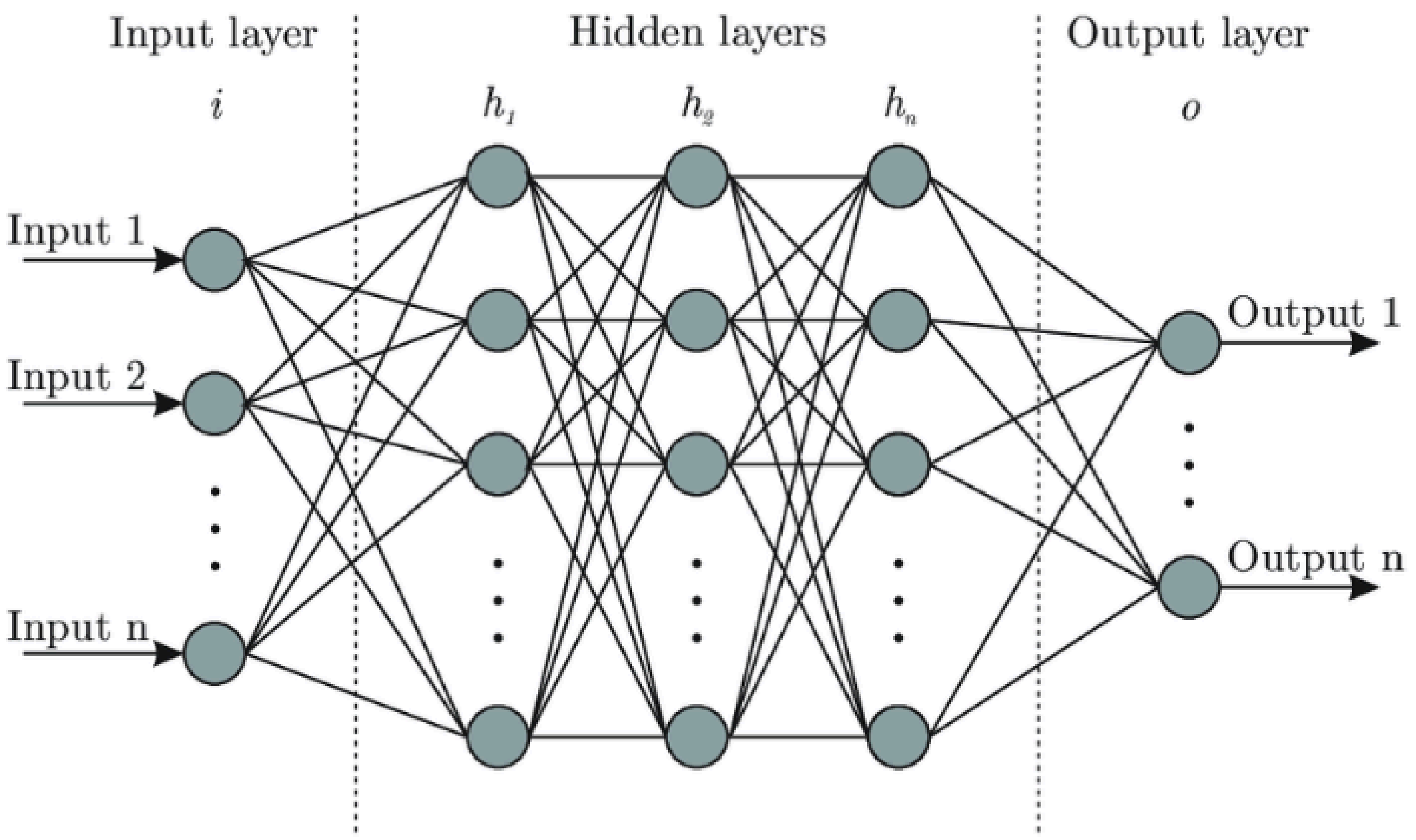


Model

Neural network

- Modify the structure of the Neural network model

1. Add a hidden layer
2. Modify the number of nodes per laye

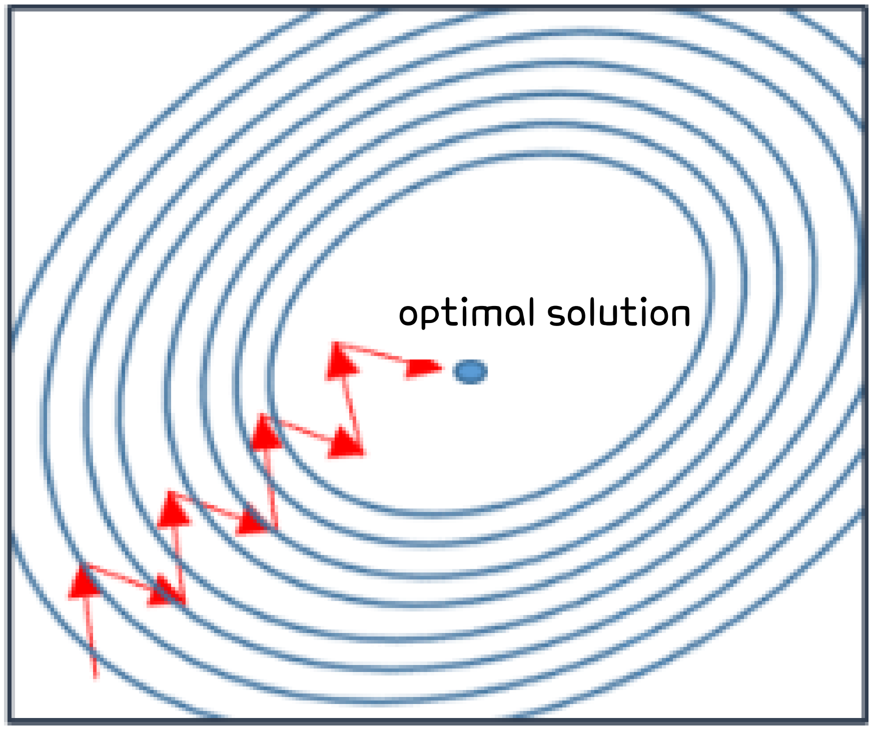


Optimization

SGD(Stochastic Gradient Descent)

- Adjust weights for some randomly extracted data

$$W(t+1) = W(t) - \alpha \frac{\partial}{\partial w} Cost(w)$$



- AdaGrad

:Adaptively adjust learning rates for each feature

➜ Automatically adjust learning rates throughout the learning process

WHY?

- Each feature is different in importance and size
- Applying the same learning rate to all features ➜ inefficient !

$$g_t = g_{t-1} + (\nabla f(x_{t-1}))^2$$

$$x_t = x_{t-1} - \frac{\eta}{\sqrt{g_t + \epsilon}} \cdot \nabla f(x_{t-1})$$

ϵ: small values added for numerical stability

η : Hyperparameters indicating the initial learning rate

Optimization

parameters

- Adam

loss='categorical_crossentropy', optimizer=opt, metrics=['accuracy']

- Adagrad

loss='categorical_crossentropy', optimizer=opt_adagrad, metrics=['accuracy'])

- SGD

loss='categorical_crossentropy', optimizer=opt_sgd, metrics=['accuracy']

numAMRs

- Assumptions of cost per simulation

: The average lifespan of an AMR is 100000km

: The average speed of the robot is 2m/s

: k is the coefficient that makes the average speed 2m/s

Cost per simulation $= \left(\frac{\text{Machine learning total distance} \times k}{100000000}\right) \times 20000 \times n$

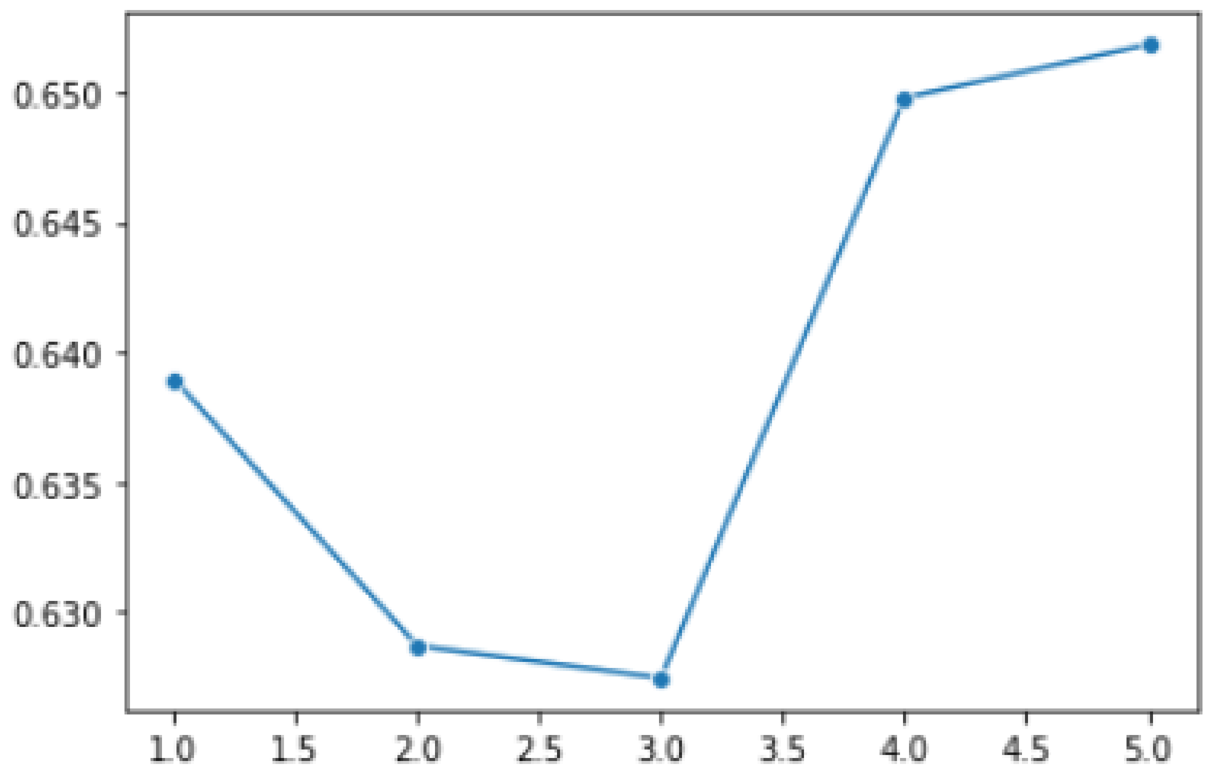
where:

- n is the number of AMRs (Autonomous Mobile Robots).
- k is the distance conversion factor.

Experimental results

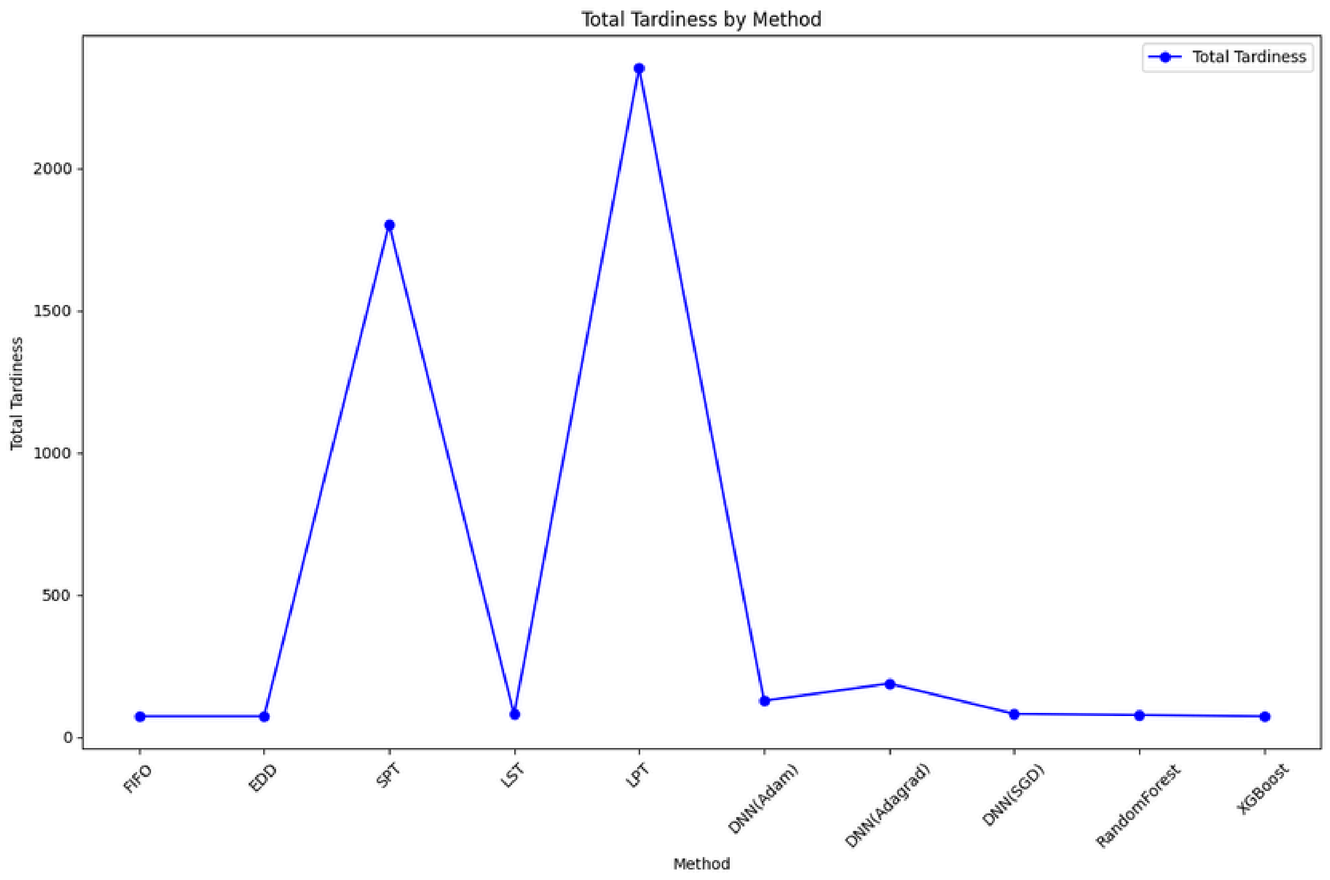
numAMRs

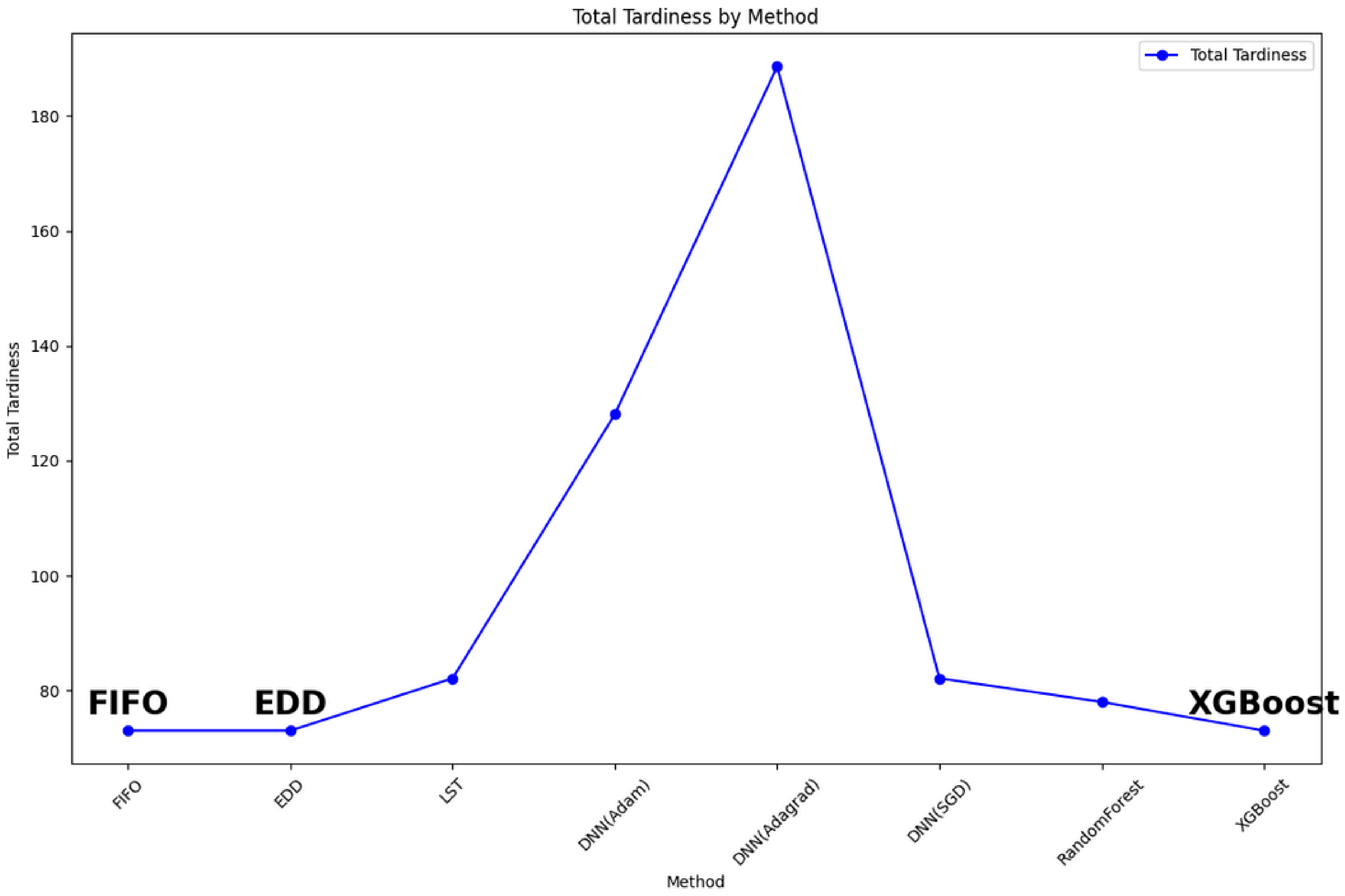
- When the number of AMRs is 3, use the lowest cost.
- Therefore, adopt 3



Total Tardiness

- After setting the number of amr to 3 ..
  : all algorithms and machine learning methods were applied



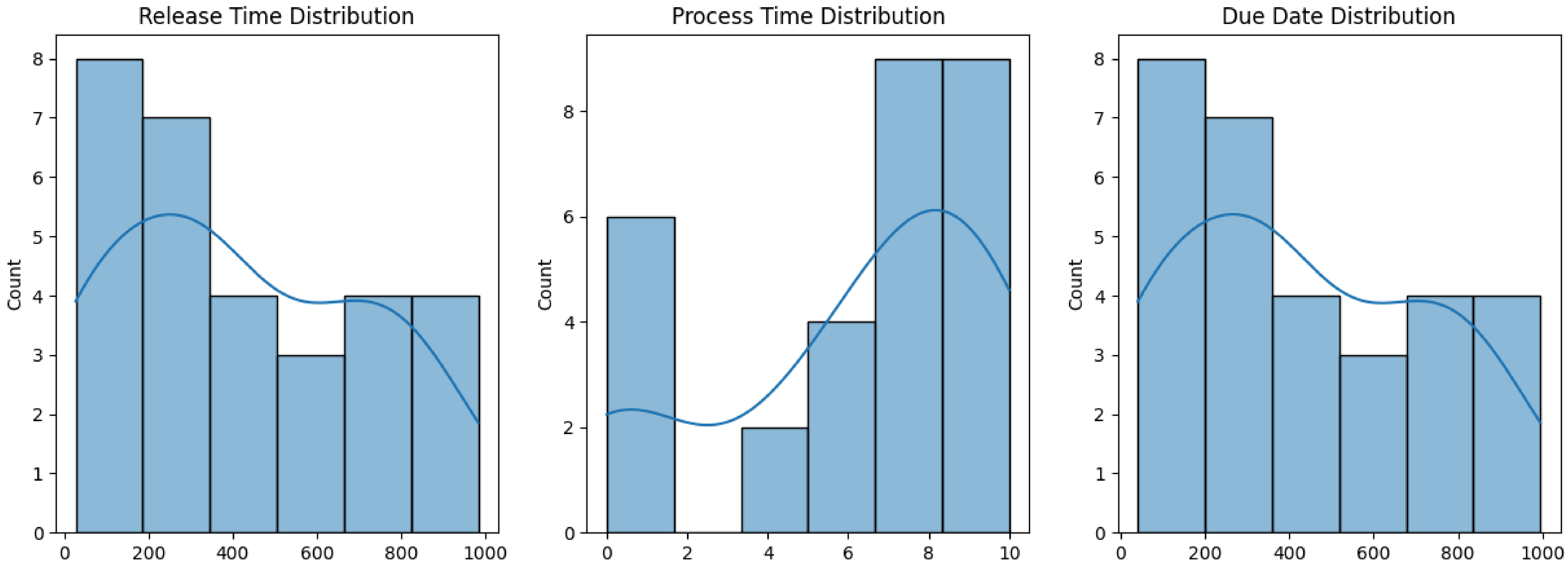


➜ In particular, FIFO, EDD, and XGBoost have low tardiness !

Conclusion

Analysis Interpretation

- Why are FIFO, EDD, and XGBoost methods effective?



Distribution of release time of material and uniform distribution of processing time and due date

- Why are FIFO, EDD, and XGBoost methods effective?

	feature1	feature2_0	feature2_1	feature2_2	feature3_0	feature3_1	feature3_2	feature4_0	feature4_1	feature4_2	response
0	0.000000	0	0	0	0.000000	0.000000	0.000000	0.0	0.0	0.0	2
1	26.248809	0	1	0	0.000000	0.000000	0.000000	27.3	14.3	0.0	1
2	21.931712	0	1	1	0.000000	0.000000	0.000000	75.7	62.7	25.7	2
3	21.470911	1	1	2	58.000000	0.000000	72.000000	11.3	114.3	5.3	0
4	21.931712	1	1	5	58.000000	0.000000	40.333333	67.1	170.1	12.1	0
...	...	...	...	...	...	...	...	...	...	...	...

The nature and quantity of datasets favor simple heuristics such as FIFO and EDD
: So, the complexity and variability of training deep neural networks such as DNNs ➜ tardiness ↑

XGBoost
: Better predictive performance by understanding the potential relationships in the data, even if the complexity or characteristics of the data are small

Desired Things

- We didn't take into account more variables in terms of cost function ..

Cost per simulation = $\left(\frac{\text{Machine learning total distance} \times k}{100000000}\right) \times 20000 \times n$

where:

- n is the number of AMRs (Autonomous Mobile Robots).
- k is the distance conversion factor.

Considered: Average life, speed, distance, number of AMRs

Actual environment: maintenance costs, energy consumption, operating costs of amr

Desired Things

- We have tried several times to increase the features and volume of the data .. BUT!

	feature1	feature2_0	feature2_1	feature2_2	feature3_0	feature3_1	feature3_2	feature4_0	feature4_1	feature4_2	feature5_0	feature5_1	feature5_2	feature6_0	feature6_1	feature6_2	response
0	49.010203	0	0	0	0.0	0.0	0.0	0.0	0.0	0.0	4.083333	4.083333	4.083333	0.000000	0.000000	0.000000	2
1	12.041595	1	1	1	17.0	17.0	17.0	14.0	14.0	14.0	4.090909	4.090909	4.090909	0.000000	0.000000	0.000000	2
2	0.000000	0	0	0	0.0	0.0	0.0	0.0	0.0	0.0	4.625000	4.625000	4.625000	0.000000	0.000000	0.000000	2
3	0.000000	0	0	0	0.0	0.0	0.0	0.0	0.0	0.0	5.727273	5.727273	5.727273	0.000000	0.000000	0.000000	2
4	0.000000	0	0	0	0.0	0.0	0.0	0.0	0.0	0.0	4.916667	4.916667	4.916667	0.000000	0.000000	0.000000	2
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
995	0.000000	0	0	0	0.0	0.0	0.0	19.0	19.0	19.0	3.400000	3.400000	3.400000	19.875148	19.875148	19.875148	1
996	0.000000	0	0	0	24.0	24.0	24.0	10.1	10.1	10.1	3.727273	3.727273	3.727273	25.661379	25.661379	25.661379	1
997	0.000000	0	0	0	0.0	0.0	0.0	76.3	76.3	76.3	6.400000	6.400000	6.400000	23.248353	23.248353	23.248353	1
998	0.000000	0	0	0	0.0	0.0	0.0	0.0	0.0	0.0	4.545455	4.545455	4.545455	22.027120	22.027120	22.027120	1
999	0.000000	0	0	0	0.0	0.0	0.0	2.8	2.8	2.8	4.125000	4.125000	4.125000	18.930238	18.930238	18.930238	1

1000 rows × 17 columns

No significant features found.

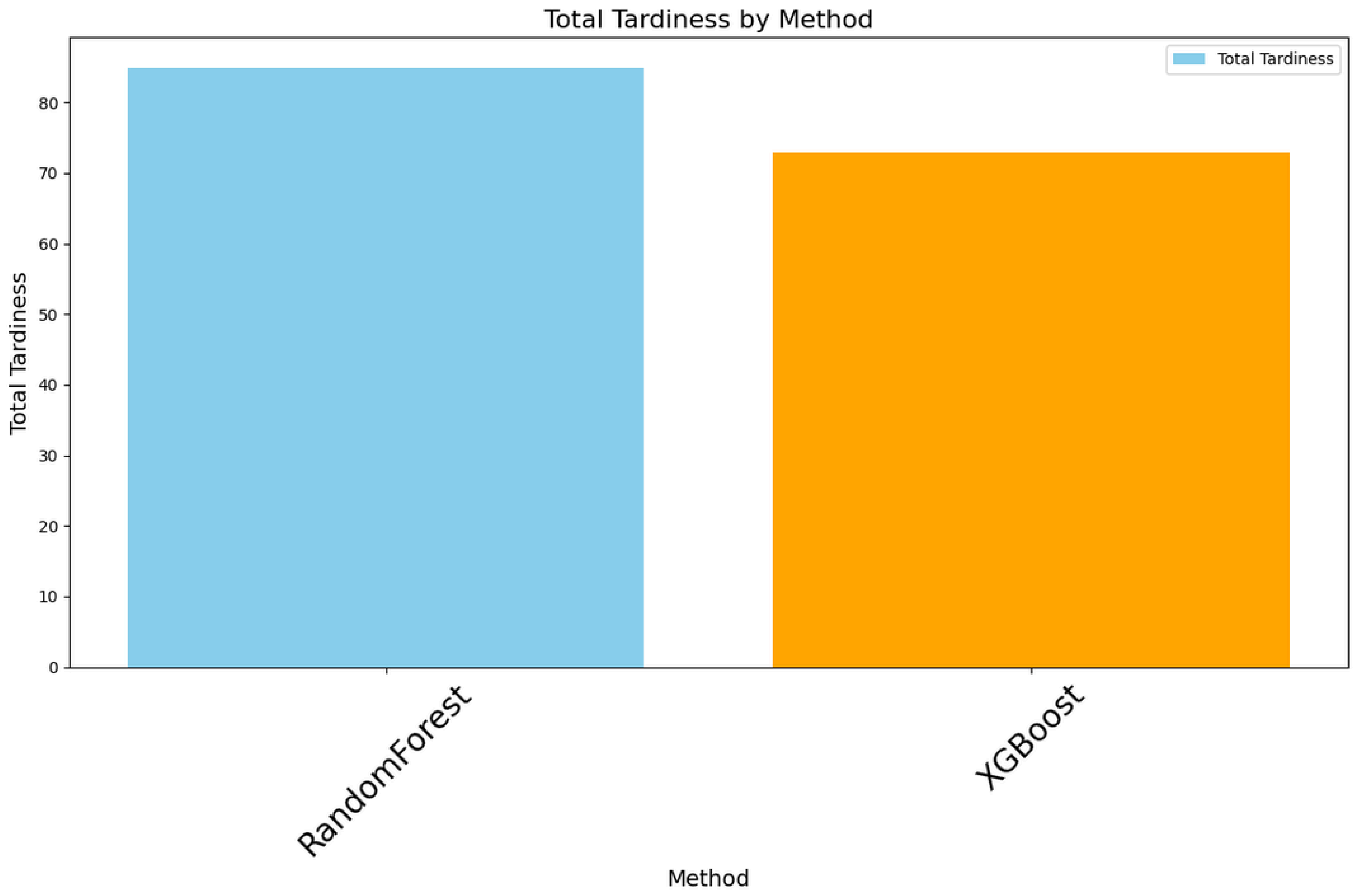
No significant change in the accuracy of the model after increasing the data.

```
totalTardiness (FIFO): 514.3
totalTardiness (EDD): 502.6999999999999
totalTardiness (SPT): 2999.7
totalTardiness (ML): 1253.8000000000002
```

```
Test Loss: 1.1258
Test Accuracy: 0.3440
```

Insight

- If you deal with similar data..



XGBoost's tardiness is lower than RandomForests'tardiness!

XGBoost is recommended when analyzing similar data.

Q&A

Thank you for listening

Reference

https://blog-ko.superb-ai.com/algorithm-in-3-minutes-gradient-boosting-model/

https://velog.io/@jjw9599/MLboostingconcept1

https://www.researchgate.net/figure/Artificial-neural-network-architecture-ANN-i-h-1-h-2-h-n-o_fig1_321259051

https://ko.wikipedia.org/wiki/%EB%9E%9C%EB%8D%A4_%ED%8F%AC%EB%A0%88%EC%8A%A4%ED%8A%B8

https://twinw.tistory.com/247

https://minsunstudio.tistory.com/36

https://cdruf.com/excellent-job-shop-scheduling-does-not-require-fancy-tools