Decision Trees for Tactic Prediction in Coq

Liao Zhang[1,3], Lasse Blaauwbroek[1,2], Bartosz Piotrowski[1,4], Cezary Kaliszyk[3,4] and Josef Urban[1]

1 Czech Technical University, Prague, Czech Republic

2 Radboud University, Nijmegen, The Netherlands

3 University of Innsbruck, Austria

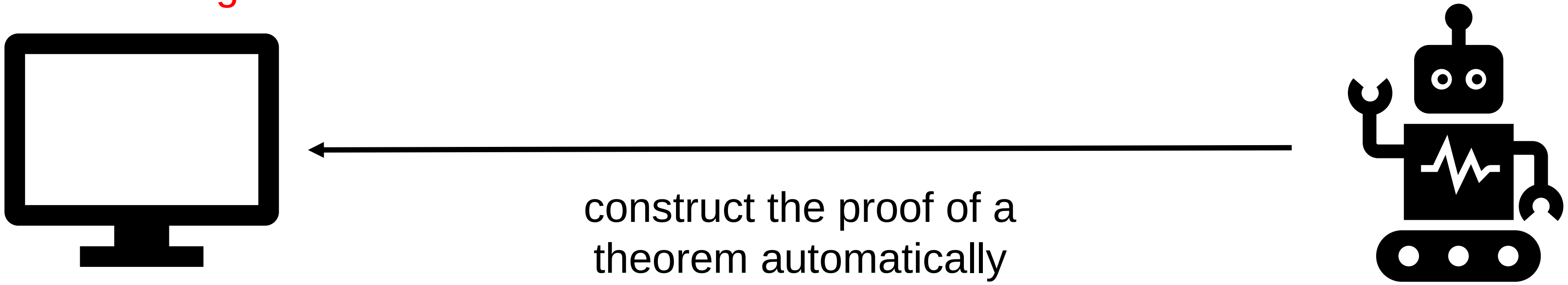
4 University of Warsaw, Poland

Introduction

- Proof automation: Coq Tactician
- Better learning model of decision trees
- More precise tactic characterization

Proof Automation

- very large tactic space
- time-consuming

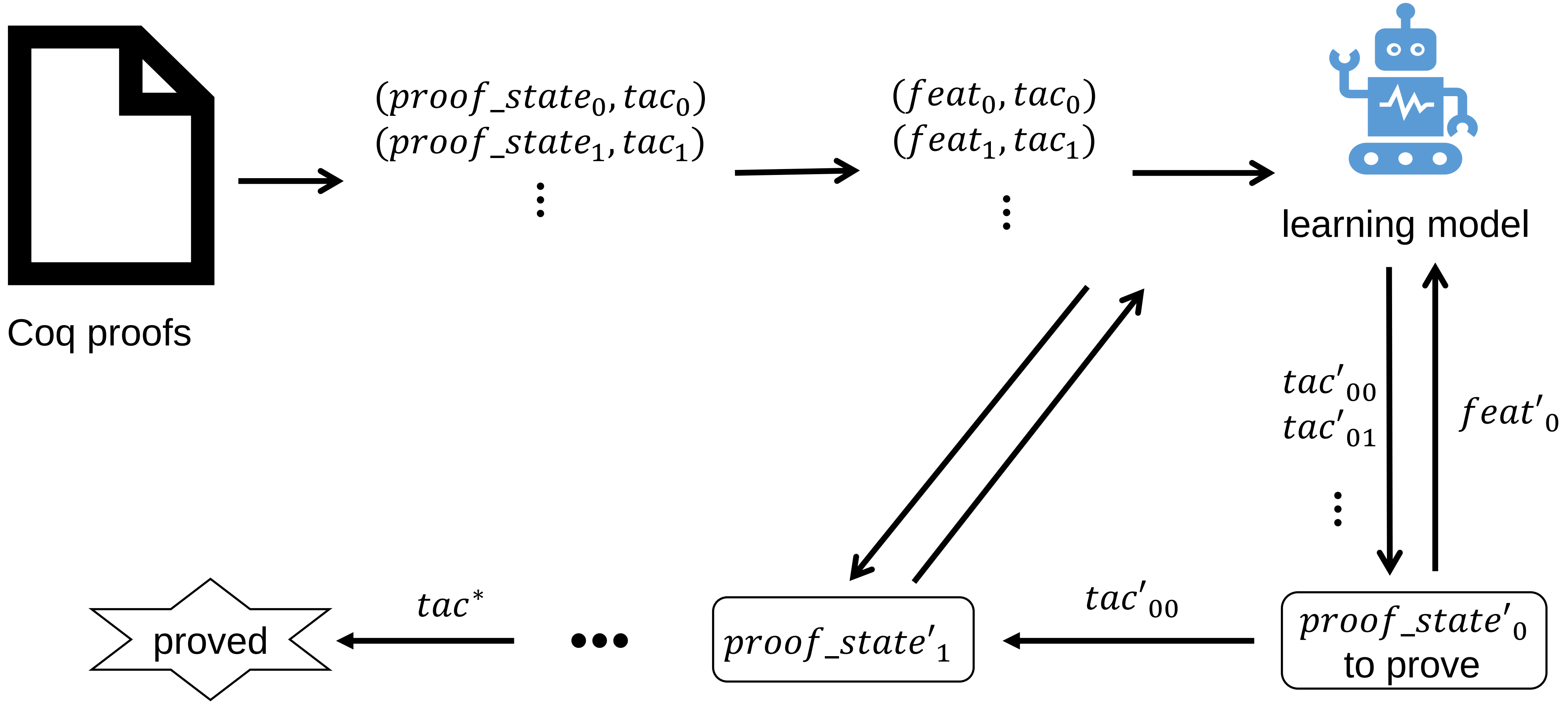


machine-mechanized proofs

machine learning

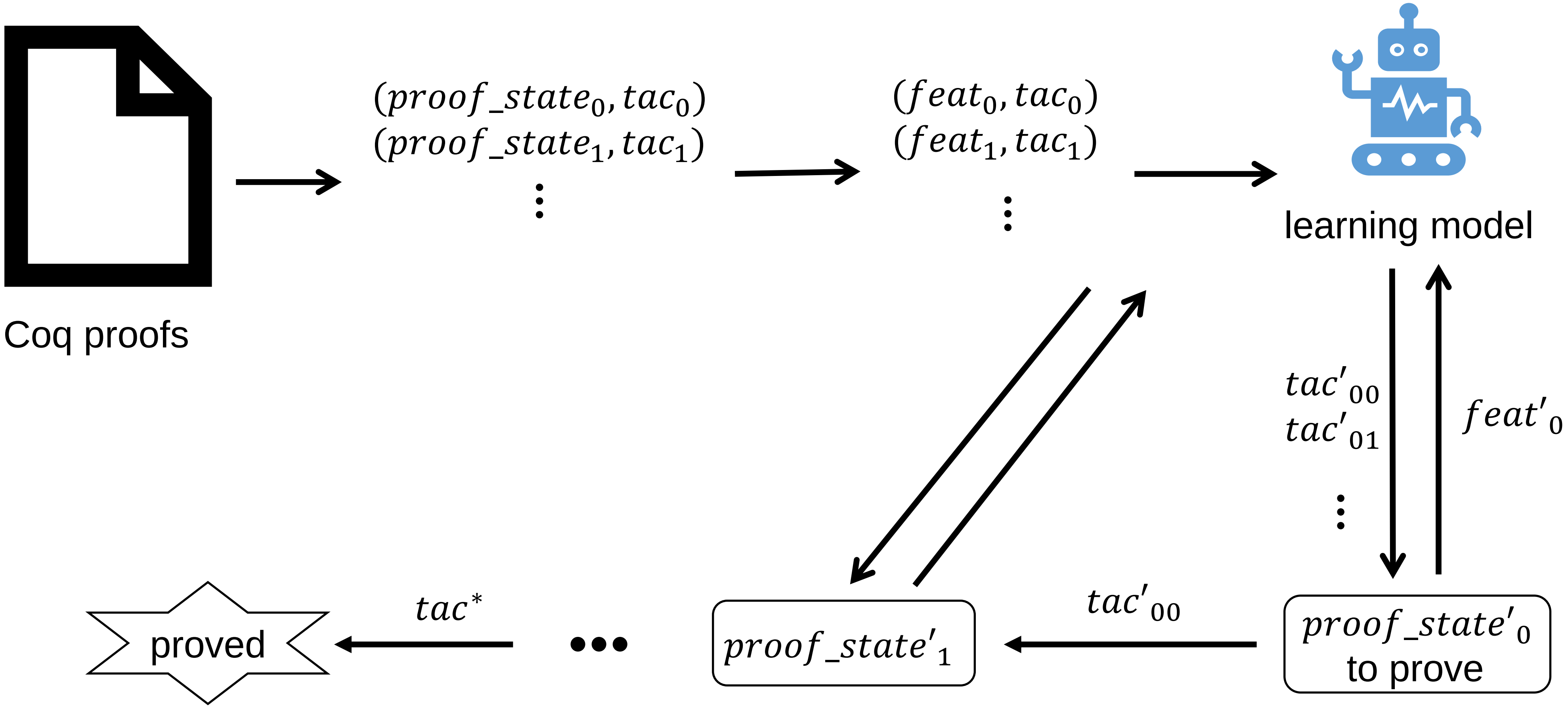
consider the proof automation for Coq

Coq Tactician



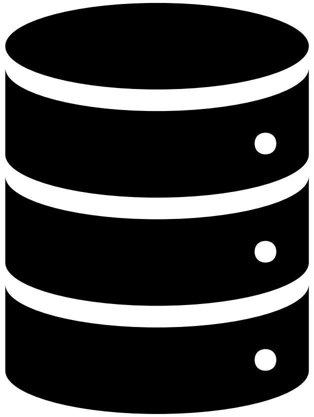
Coq Tactician

better decision tree models



Features

- Atom nodes
- Term tree walks up to length two

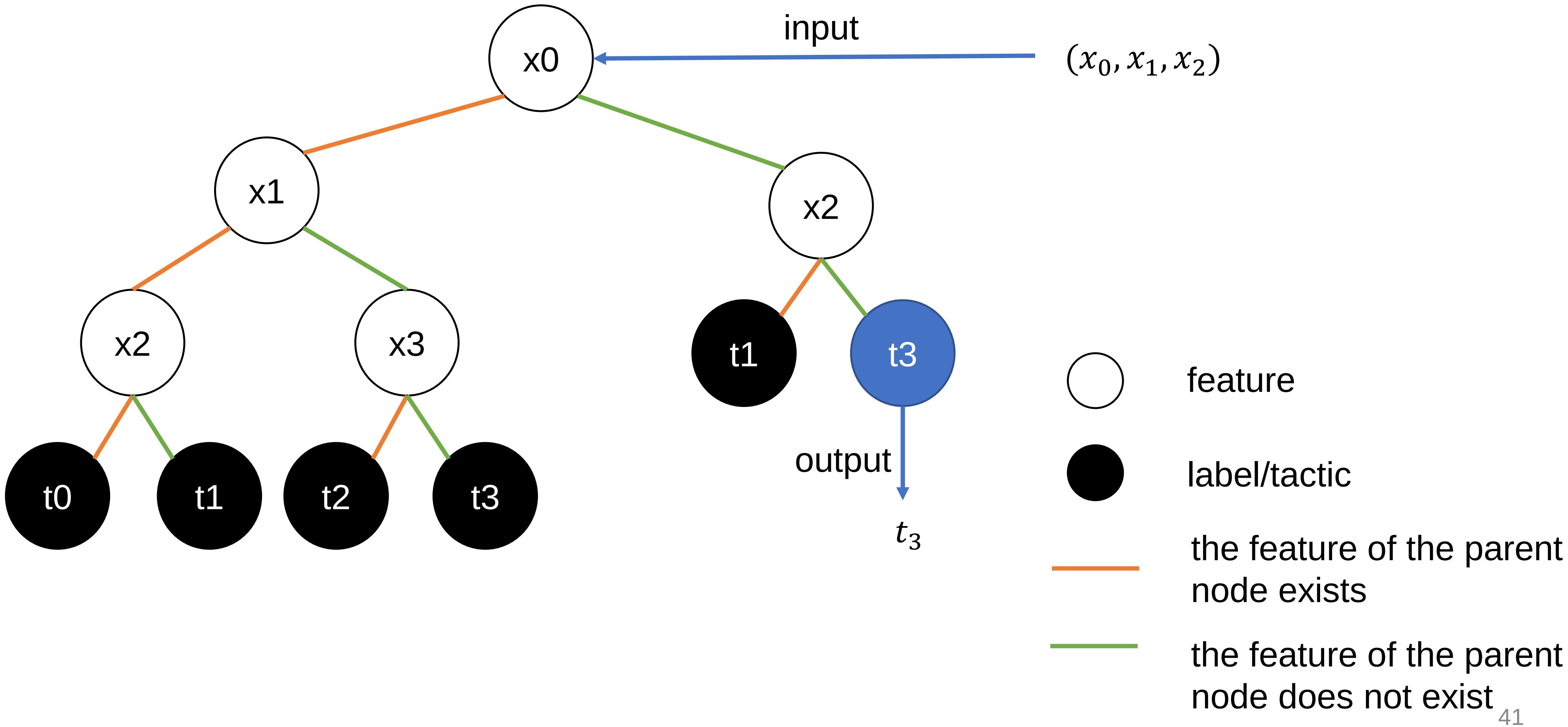


$$\{(features_i, tactic_i)\}_{i \in 1 \ldots n}$$

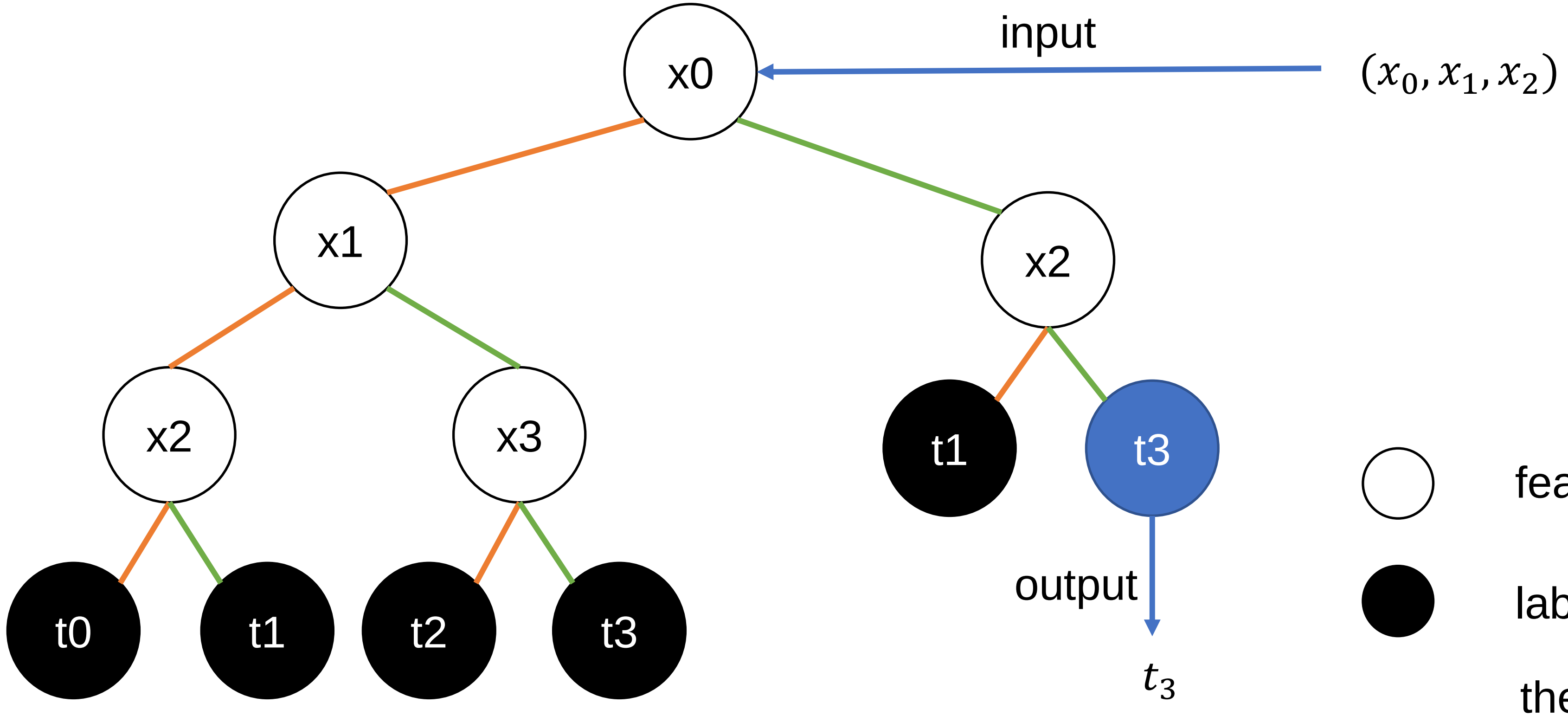
K-Nearest Neighbors (*k*-NN)

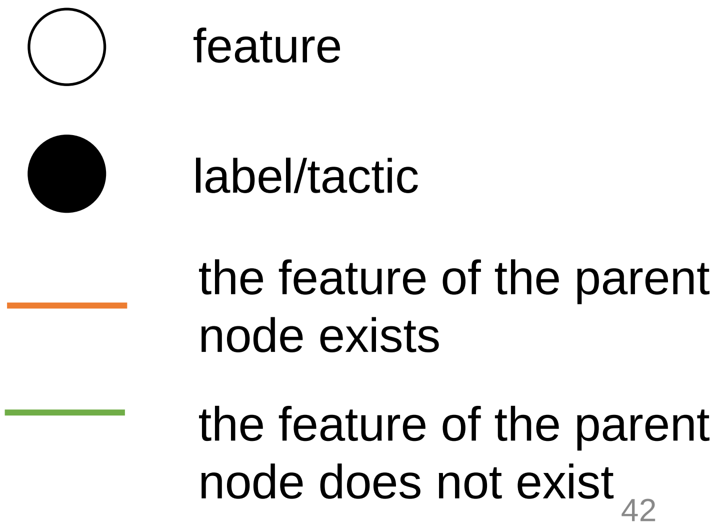
- *K*-NN:
  - Originally used in Tactician
  - Sort likely tactics by the distance measurement between proof states
- Very weak learner:
  - a stronger learner — decision trees

Decision Trees



Decision Trees

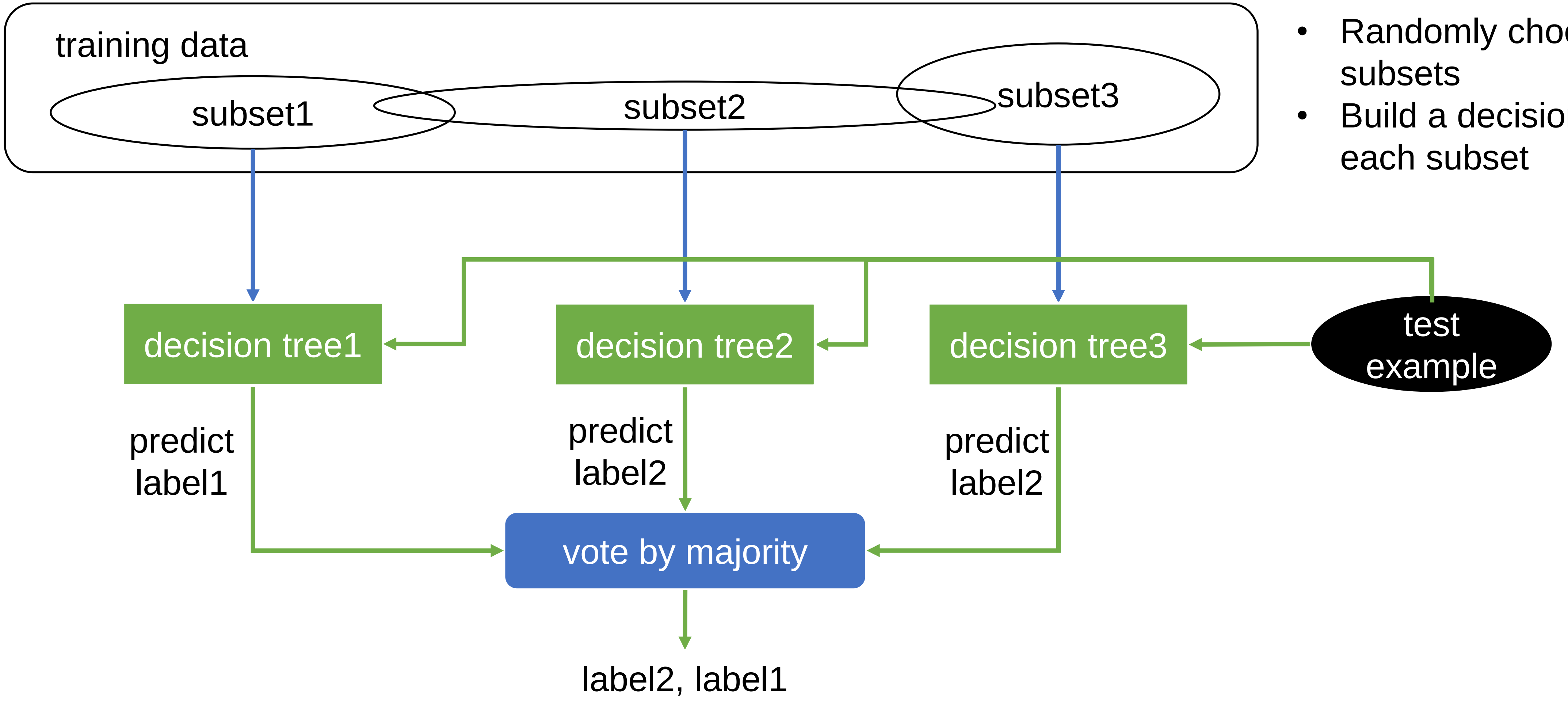




Easy to overfit:

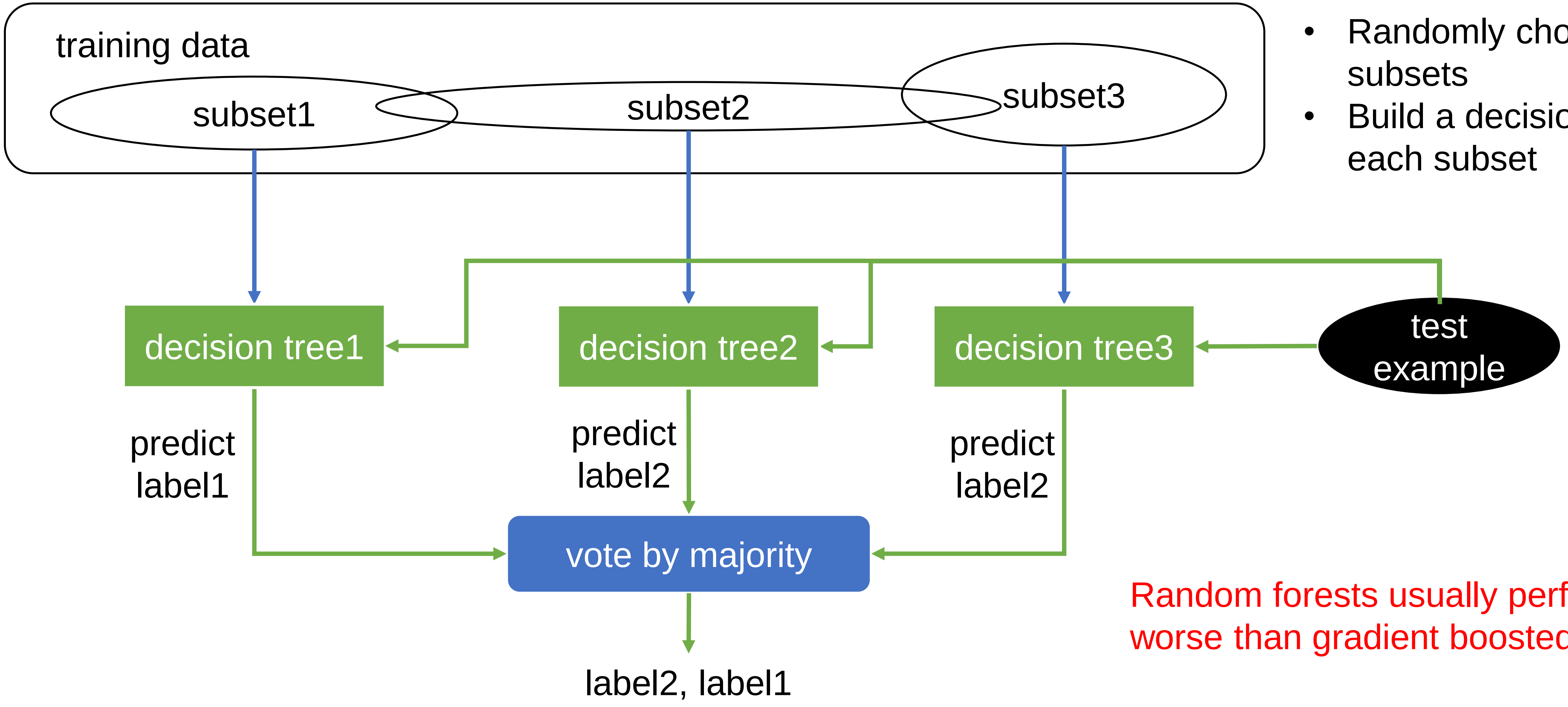
- A tree can grow very deep
- Tend to learn highly irregular patterns

Random Forests (RF)



- Randomly choose some subsets
- Build a decision tree for each subset

Random Forests (RF)



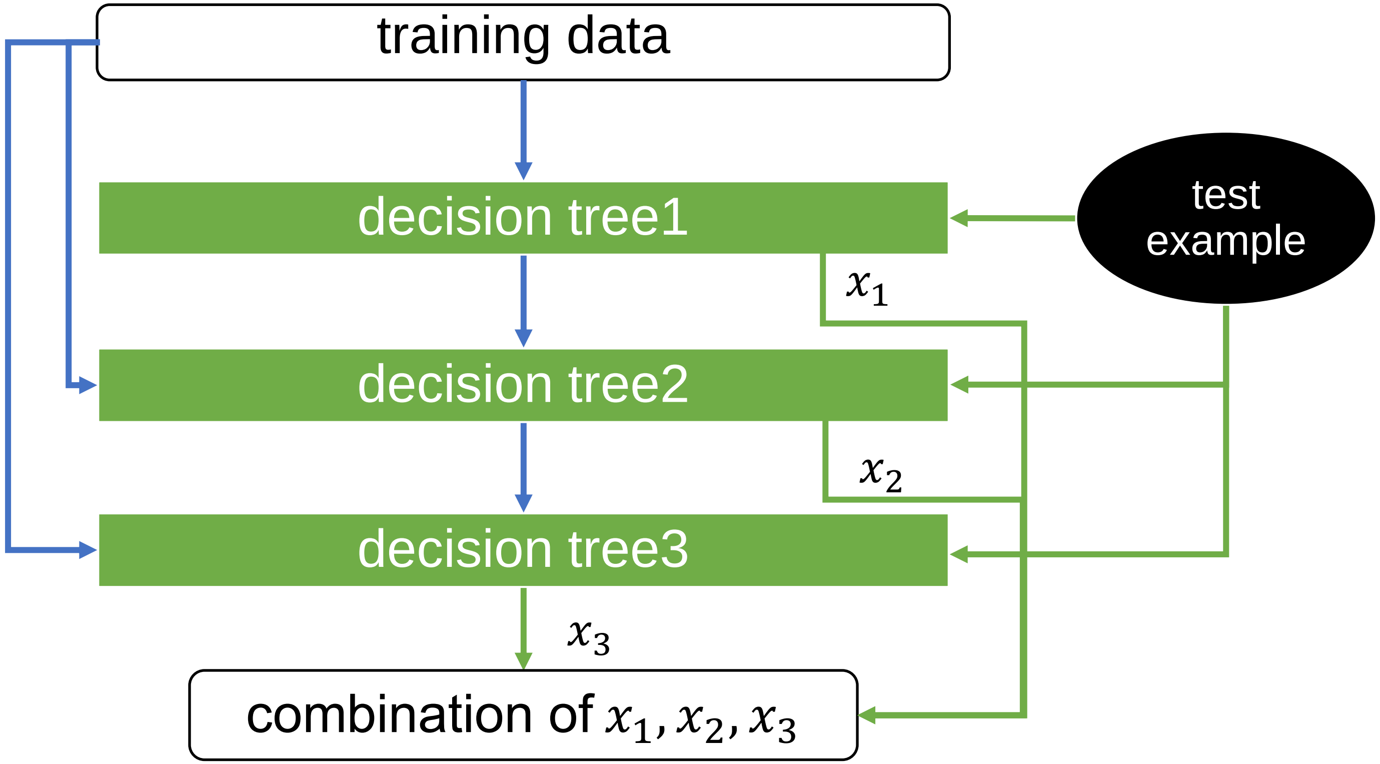
- Randomly choose some subsets
- Build a decision tree for each subset

Random forests usually perform worse than gradient boosted trees

Gradient Boosted Trees

Training:

- Build several decision trees
- The next decision tree minimizes the mistake made by the previous trees



Models

- Classification
  - Categorize a given set of data (state features) into disjoint classes (tactics)
  - Suit the tactic prediction
- Regression
  - Output continuous values, e.g., $0 \leq value \leq 1$
  - Simulate classification
    - binary regression with negative examples
    - multi-target regression

	Classification	Regression	
		Binary	Multi-target
Random forests	✓	✓	×
Gradient boosted trees	×	✓	✓

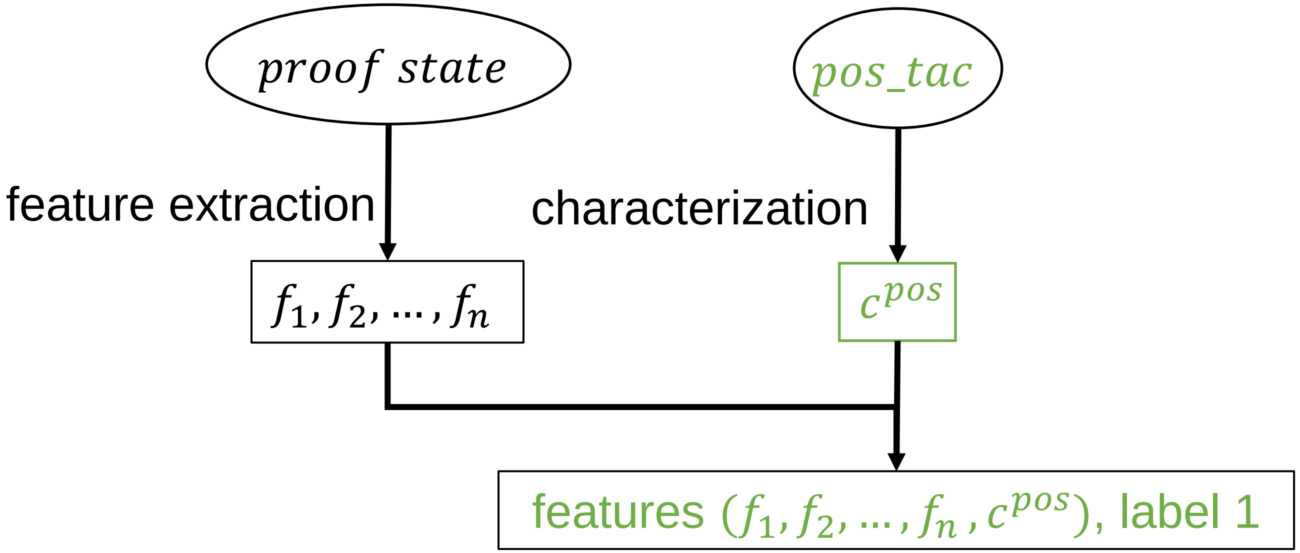
Binary Regression — Training

$proof\ state$

feature extraction

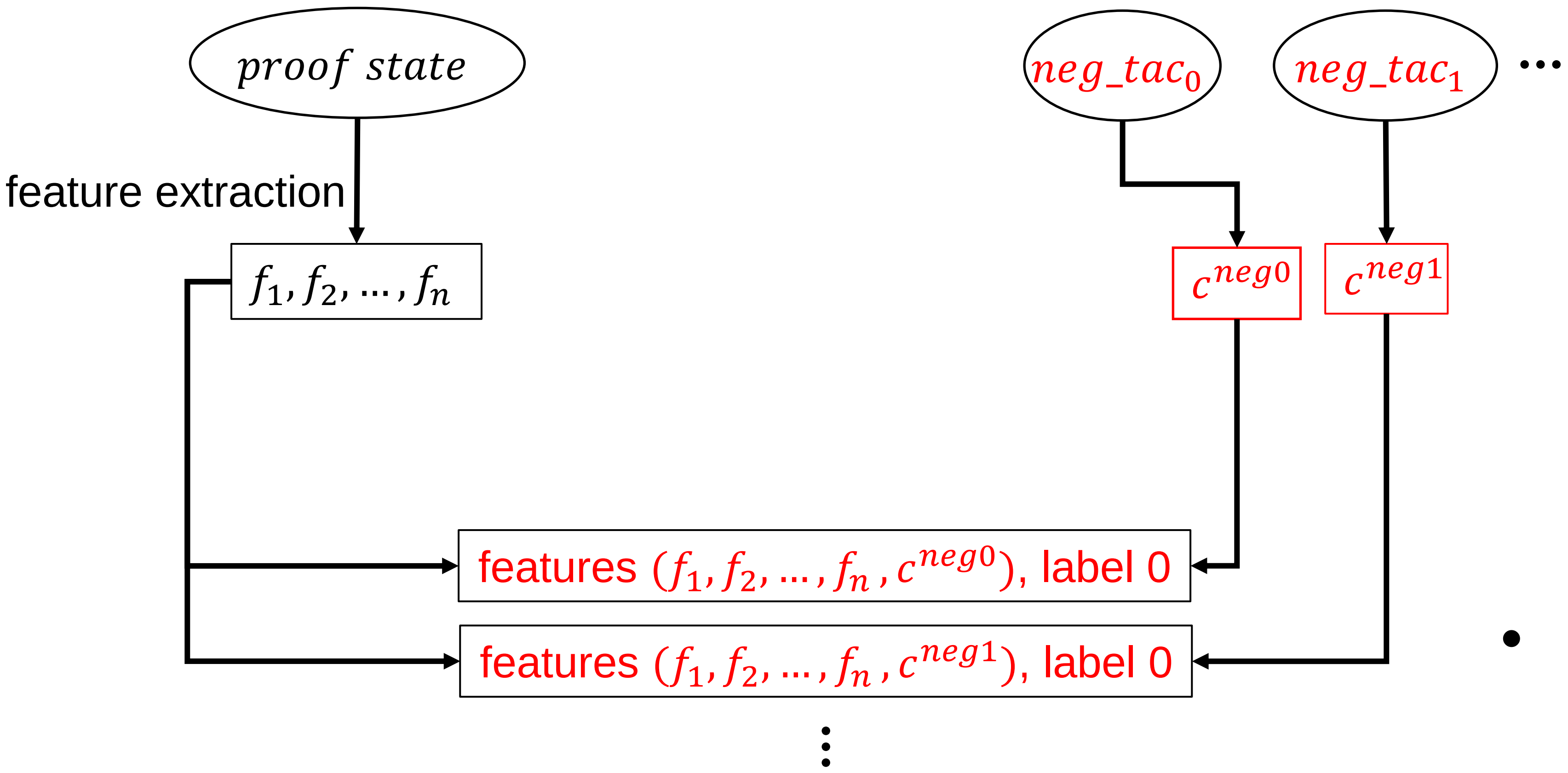
$f_1, f_2, \dots, f_n$

Binary Regression — Training



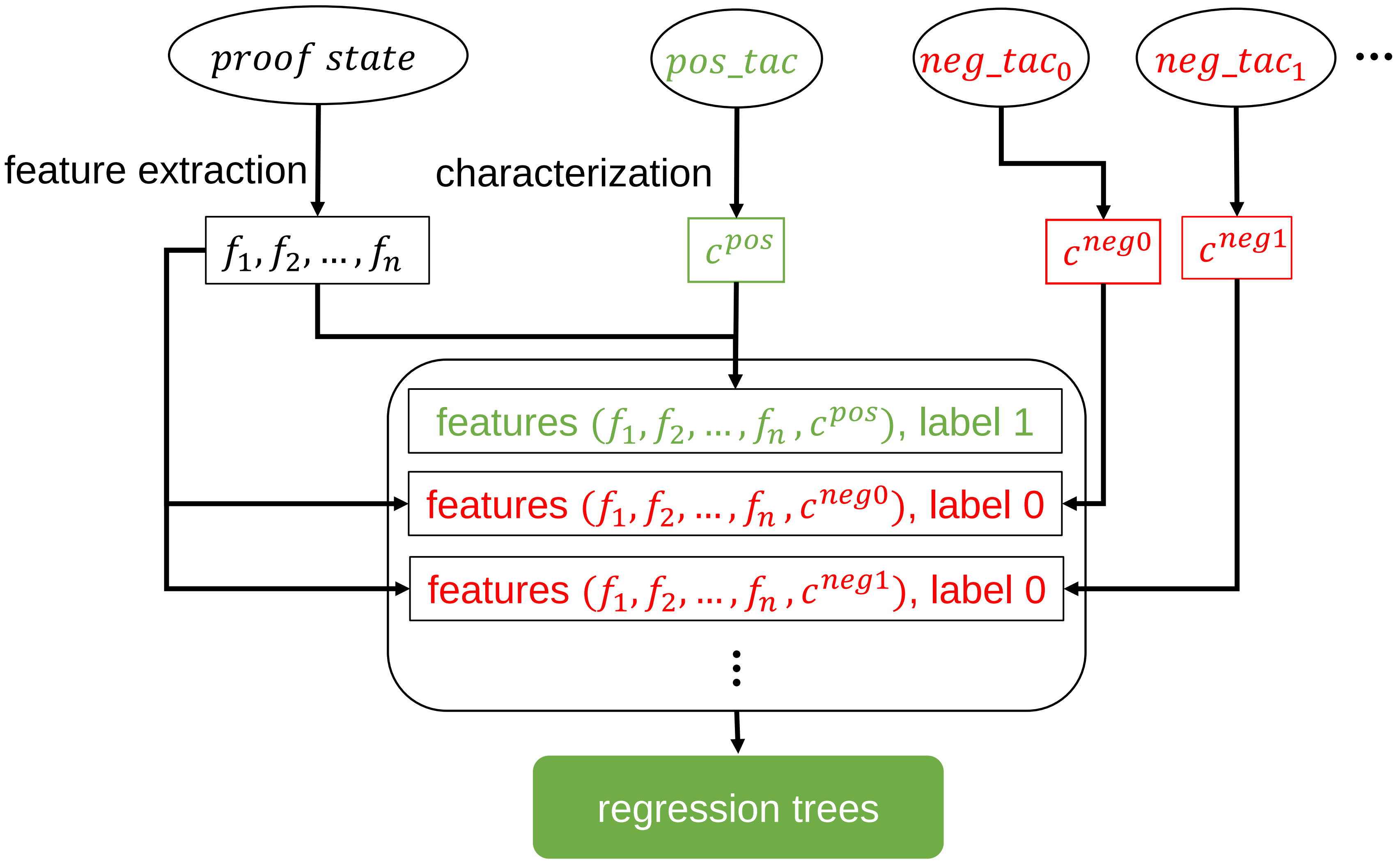
- positive tactic
  - the tactic applied to the state in the library
  - label 1

Binary Regression — Training

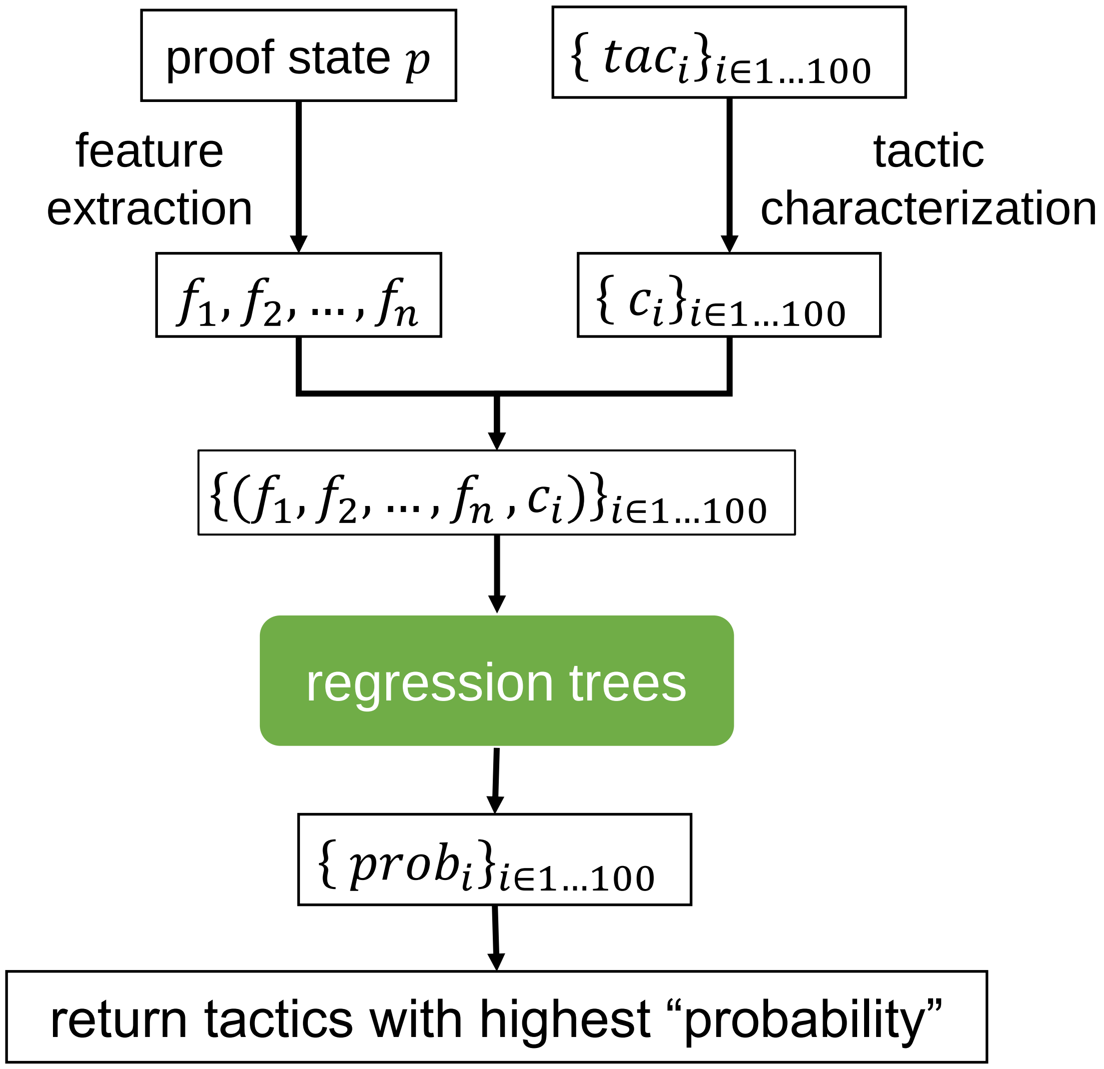


- negative tactics
  - tactics unlikely to be useful
  - label 0

Binary Regression — Training



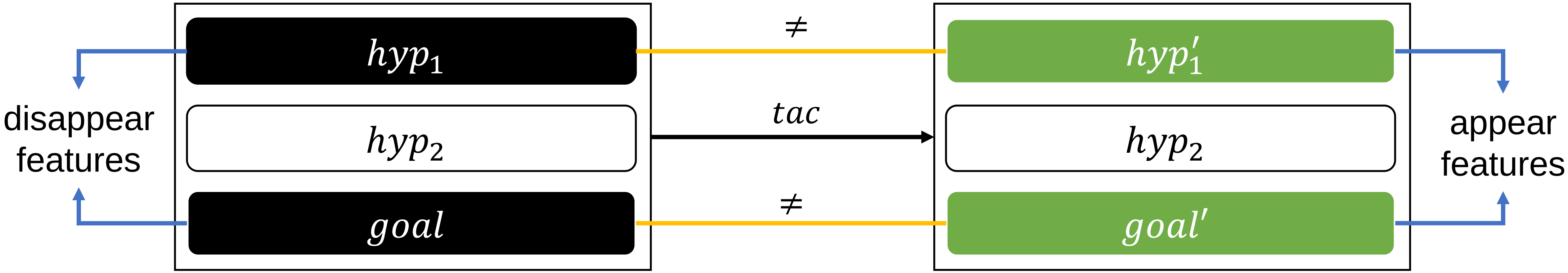
Binary Regression — Prediction



Preselect $tac_1, tac_2, \dots, tac_{100}$ maybe helpful for proving p

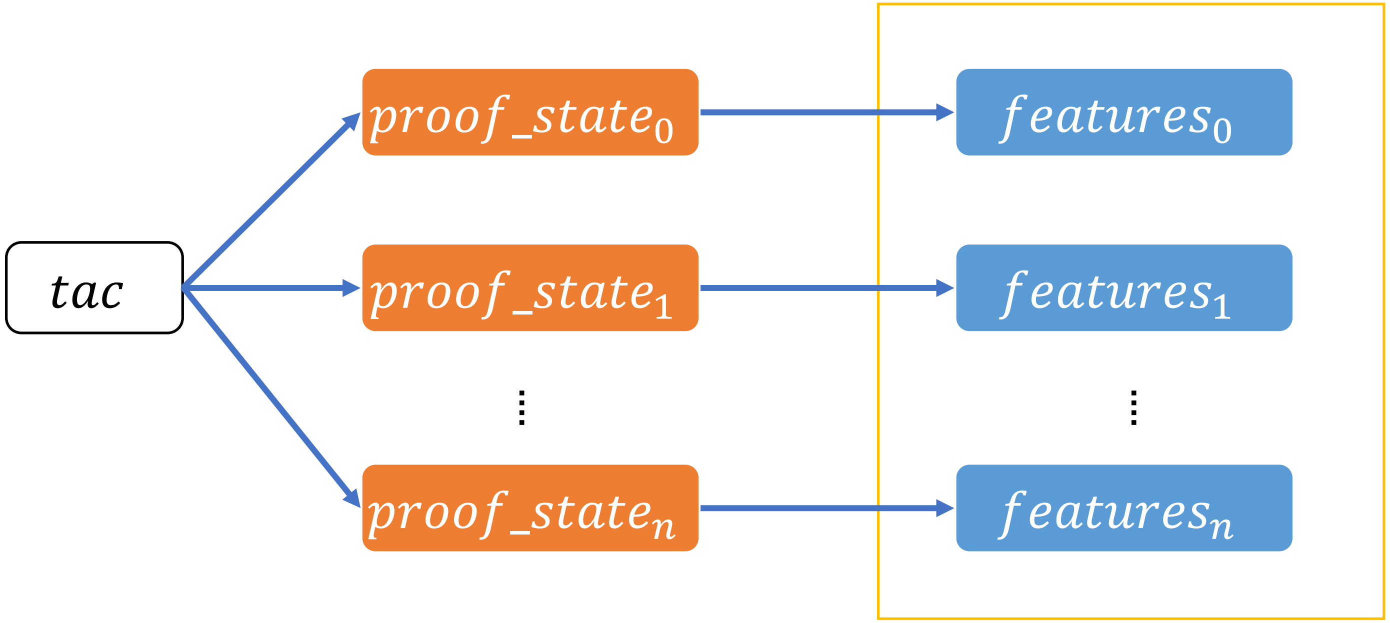
Tactic Characterization(1/2)

- Naive hash
- Features of the changed parts of the proof state
  - disappear features + appear features



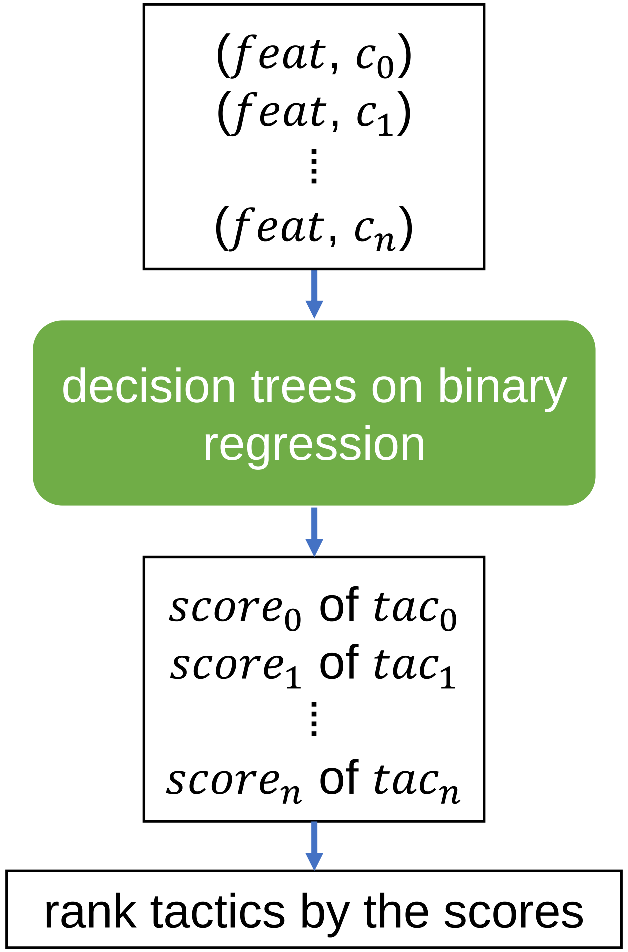
Tactic Characterization(2/2)

- Naive hash
- Features of changed parts of the proof state
- Features of all the before states

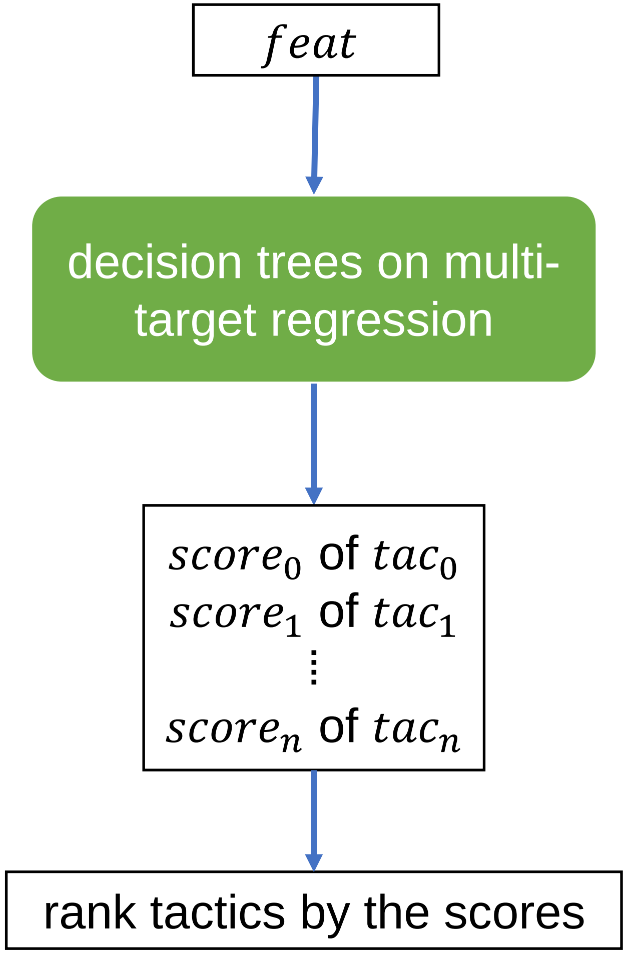


the union as the characterization

Binary/Multi-target Regression



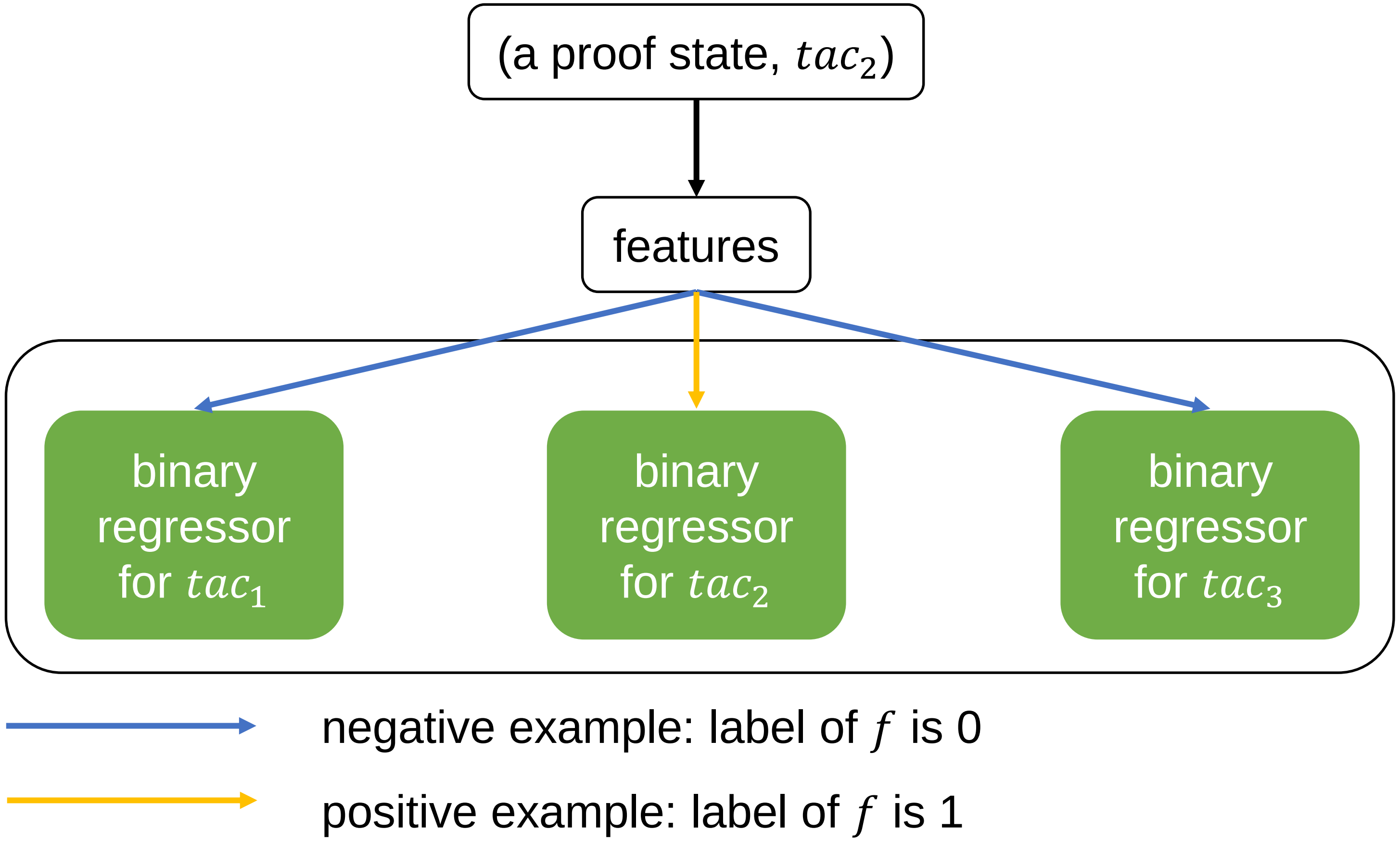
characterize tactics as features



apply tactics as labels

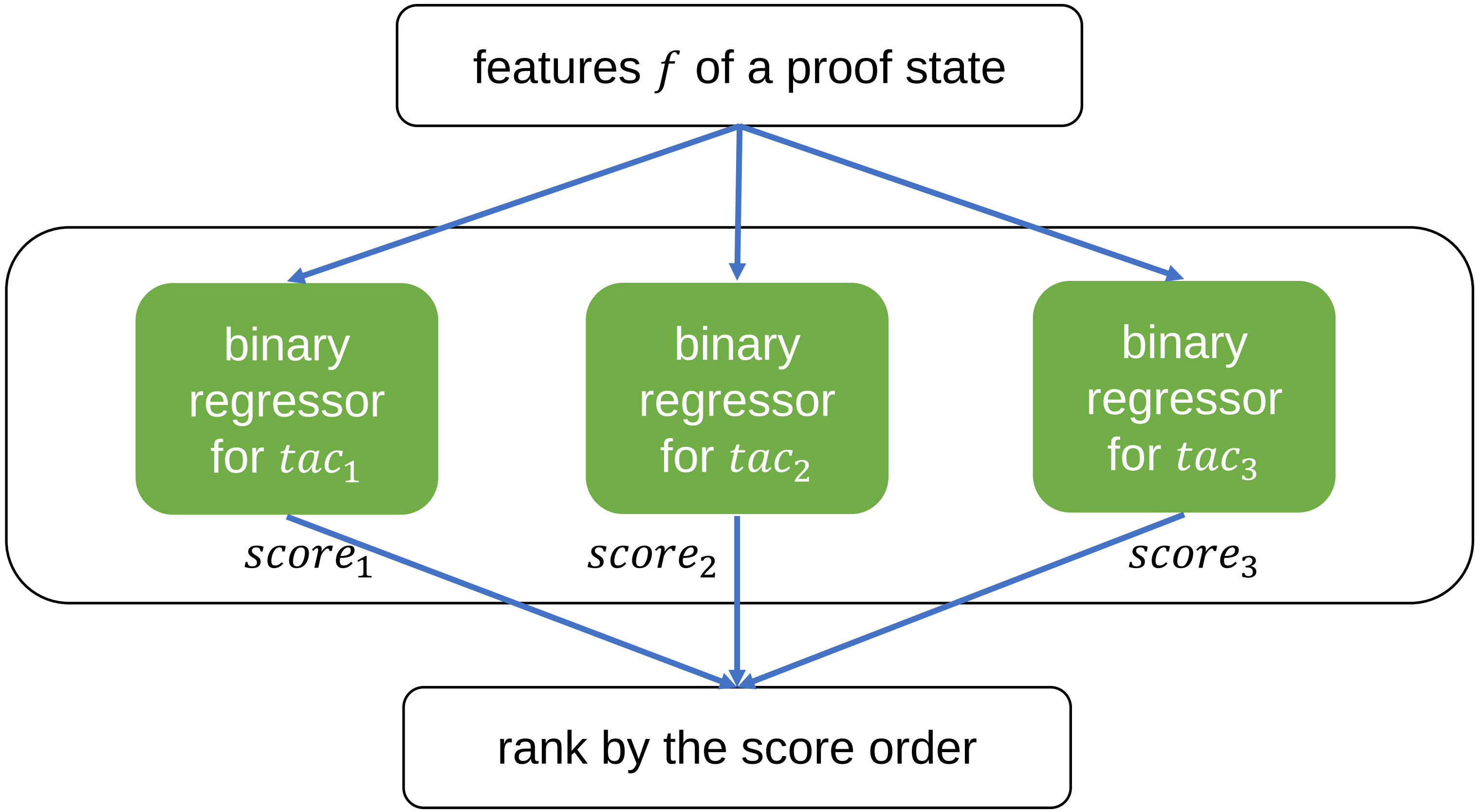
- $feat$: features of a proof state
- tac: tactics
- c_i: characterization of the tactic tac_i

Multi-target Regression — Training



- a binary regressor for each tactic
- tac_1, tac_2, tac_3 in the library
- as a positive example for the corresponding regressor
- as negative examples for the others

Multi-target Regression — Prediction

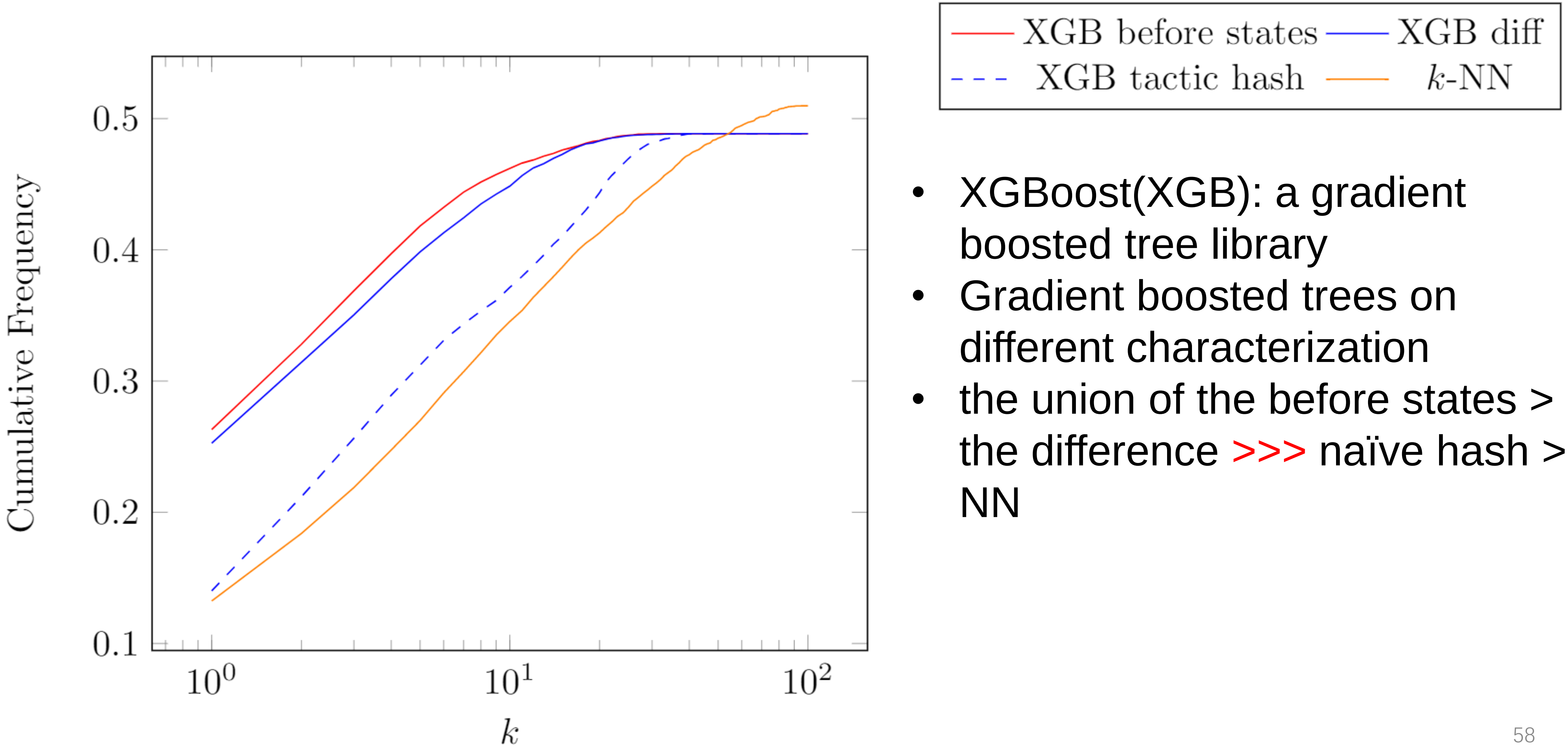


Predict a score of the corresponding tactic by each regressor

Experimental Settings

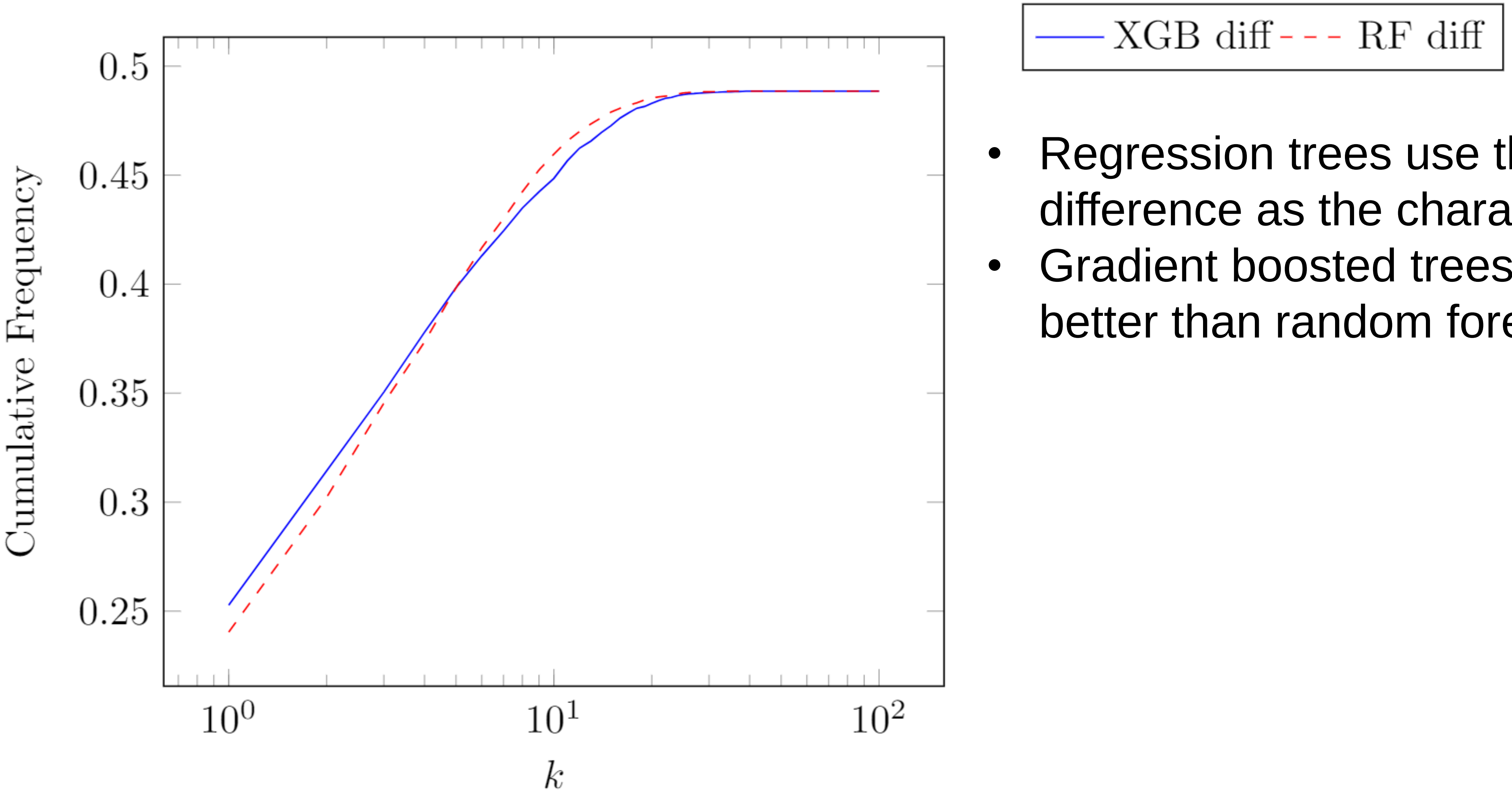
- Dataset: a random subset of the Coq standard library
- Cumulative frequency: how often the tactic in the library being presented in the first-k predictions
- Chronological evaluation: build a model for each state by learning from the previous states

Results (1/4)



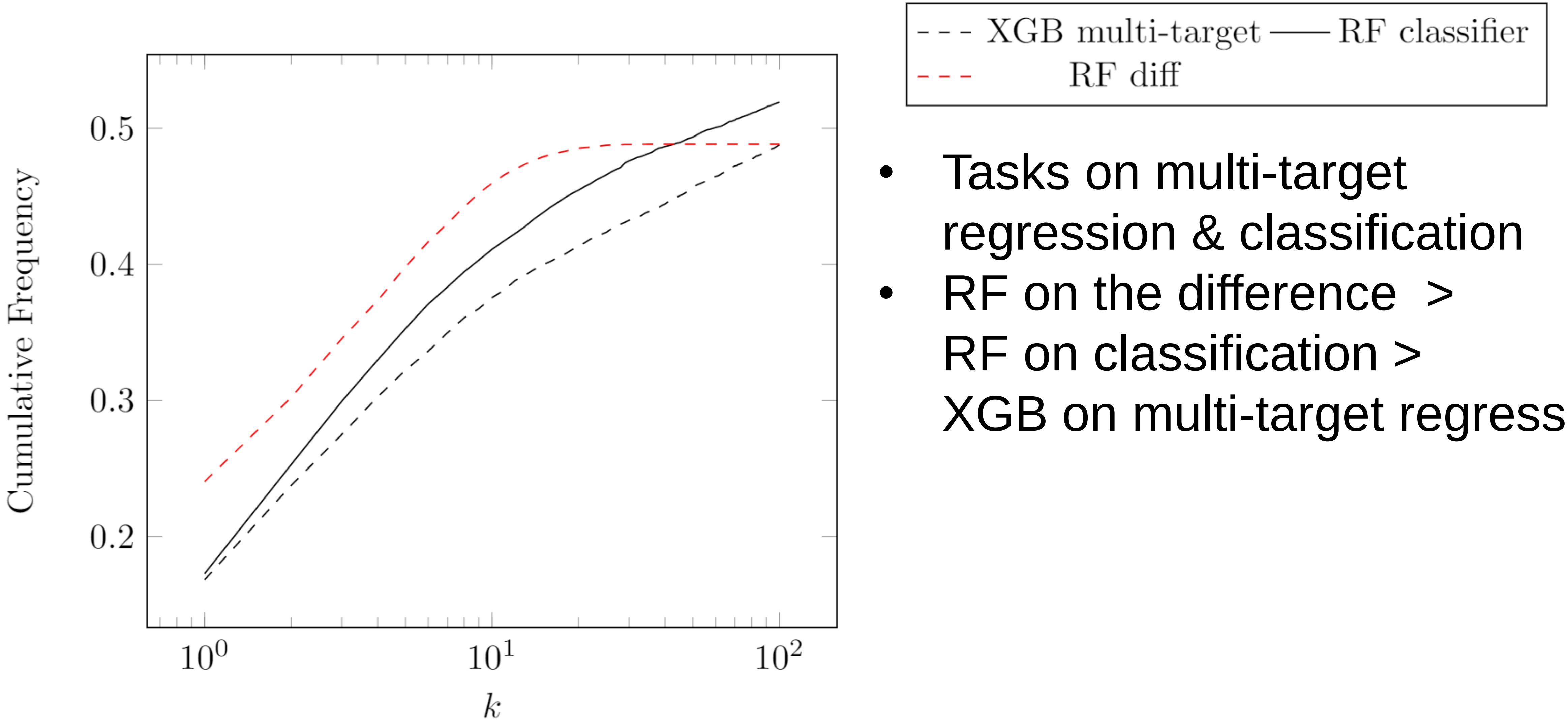
- XGBoost(XGB): a gradient boosted tree library
- Gradient boosted trees on different characterization
- the union of the before states > the difference >>> naïve hash > *k*-NN

Results (2/4)



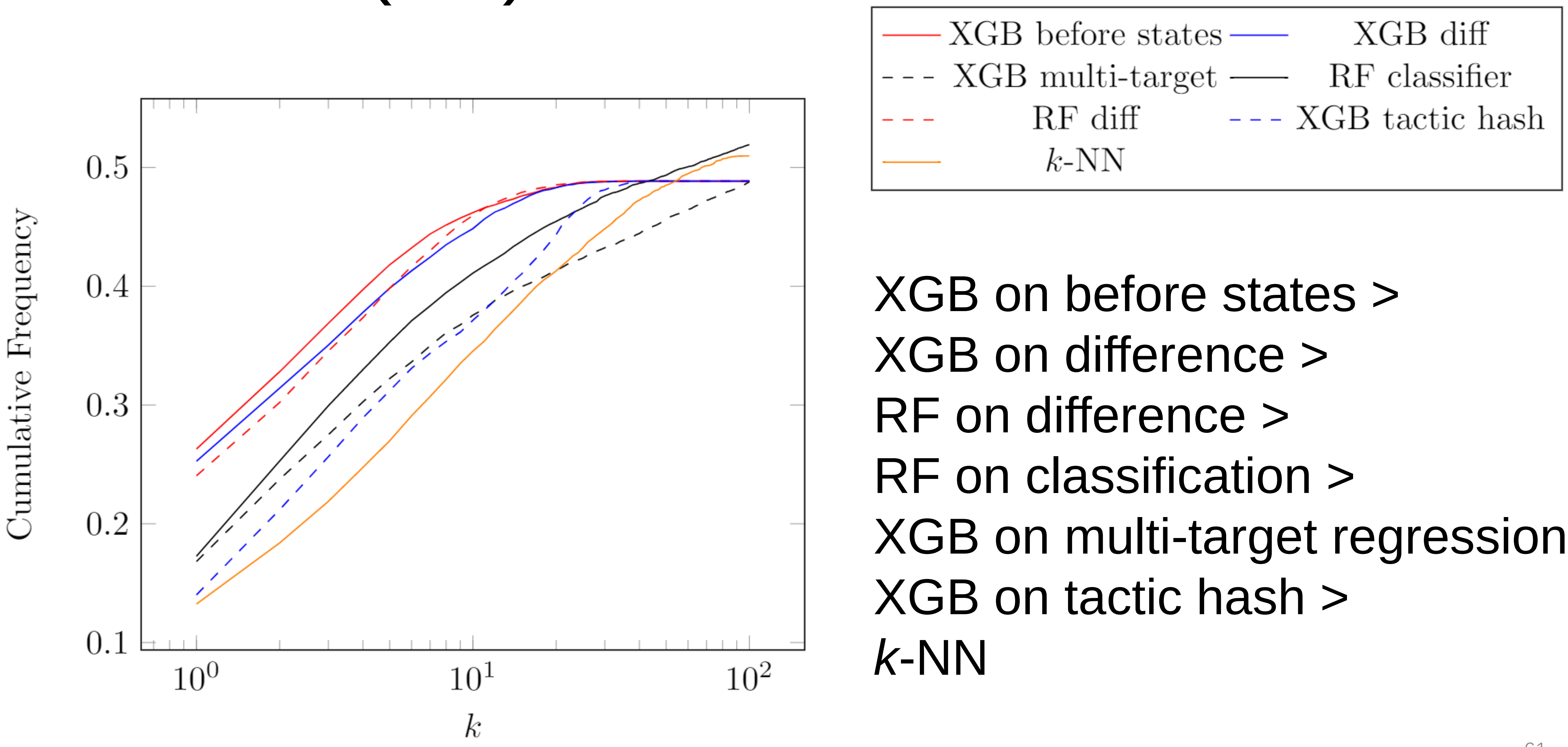
- Regression trees use the difference as the characterization
- Gradient boosted trees are a little better than random forests

Results (3/4)



- Tasks on multi-target regression & classification
- RF on the difference > RF on classification > XGB on multi-target regression

Results (4/4)



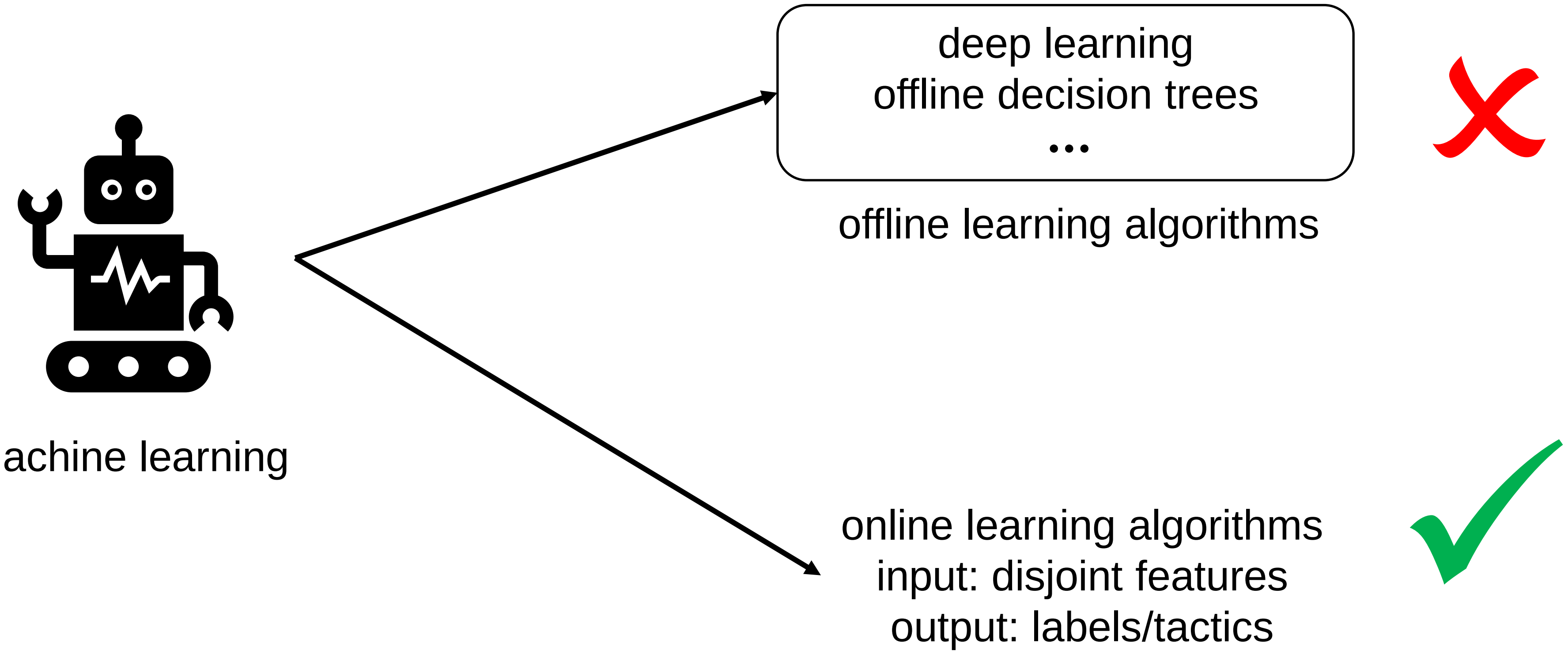
XGB on before states >
XGB on difference >
RF on difference >
RF on classification >
XGB on multi-target regression >
XGB on tactic hash >
k-NN

Conclusions and Future Work

- Conclusions:
  - Decision trees perform much better than *k*-NN on tactic prediction
  - Appropriate tactic characterization enhances the prediction power
  - No significant difference between random forests and gradient boosted trees
- Future work:
  - Improve our online random forests, see our CICM paper [Zhang etc. 2021]
  - Investigate better tactic characterization, e.g., the union of all the differences

Online Learning (1/4)

Online learning: quickly update the model after adding new training examples



Online Learning (2/4)

```
Theorem plus_O_n : forall n:nat, 0 + n = n.
Proof.
  simpl.
  reflexivity.
Qed.

Theorem mult_0_plus : forall n m : nat,
  (0 + n) * m = n * m.
Proof.
   intros n m.
   rewrite plus_O_n.
   reflexivity.
Qed.
```

```
1 subgoal
n, m : nat
______________________________________(1/1)
(0 + n) * m = n * m
```

Need the result of $plus\_O\_n$

Online Learning (3/4)

```
Theorem plus_O_n : forall n:nat, 0 + n = n.
Proof.
  simpl.
  reflexivity.
Qed.

Theorem mult_0_plus : forall n m : nat,
   (0 + n) * m = n * m.
Proof.
   intros n m.
   rewrite plus_O_n.
   reflexivity.
Qed.
```

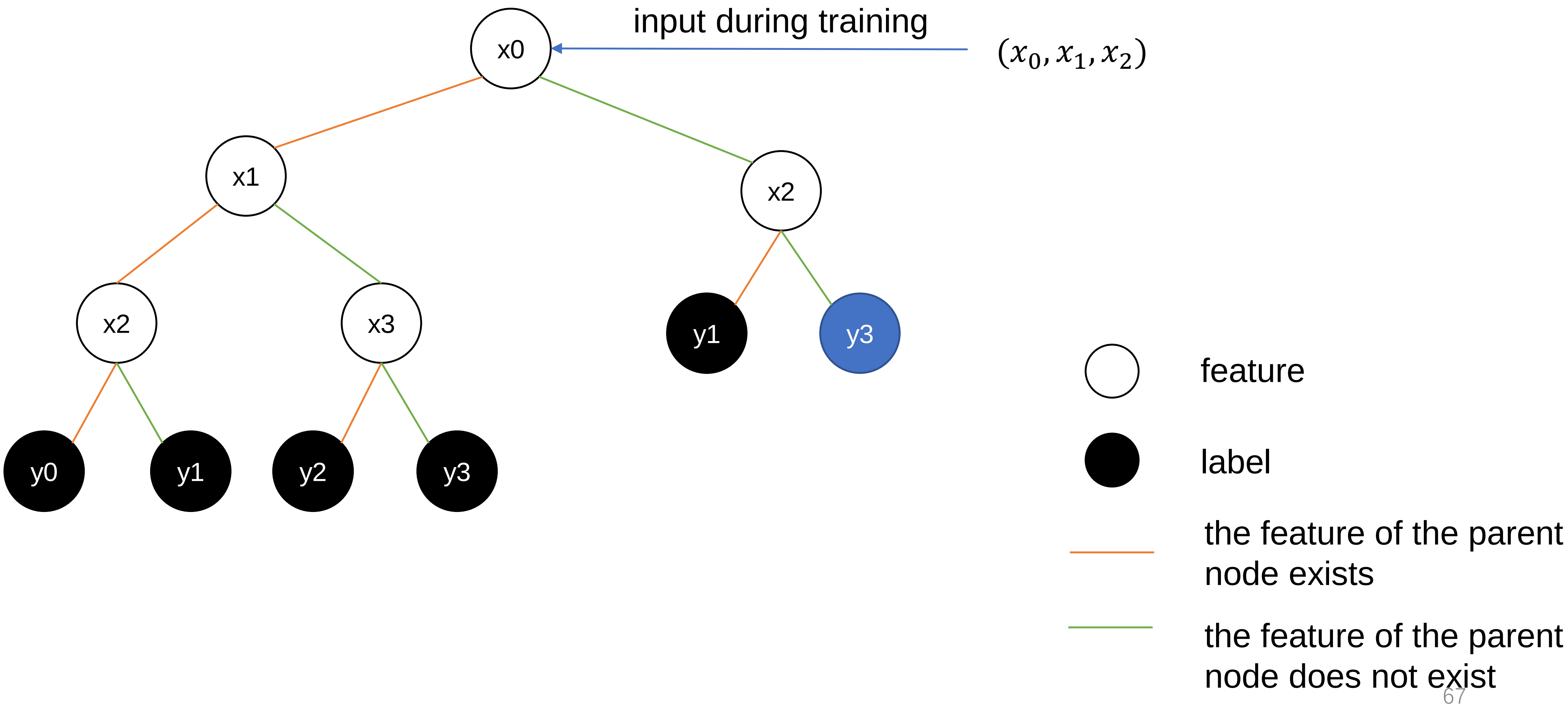
```
1 subgoal
n, m : nat
________________________________________(1/1)
n * m = n * m
```

Online Learning (4/4)

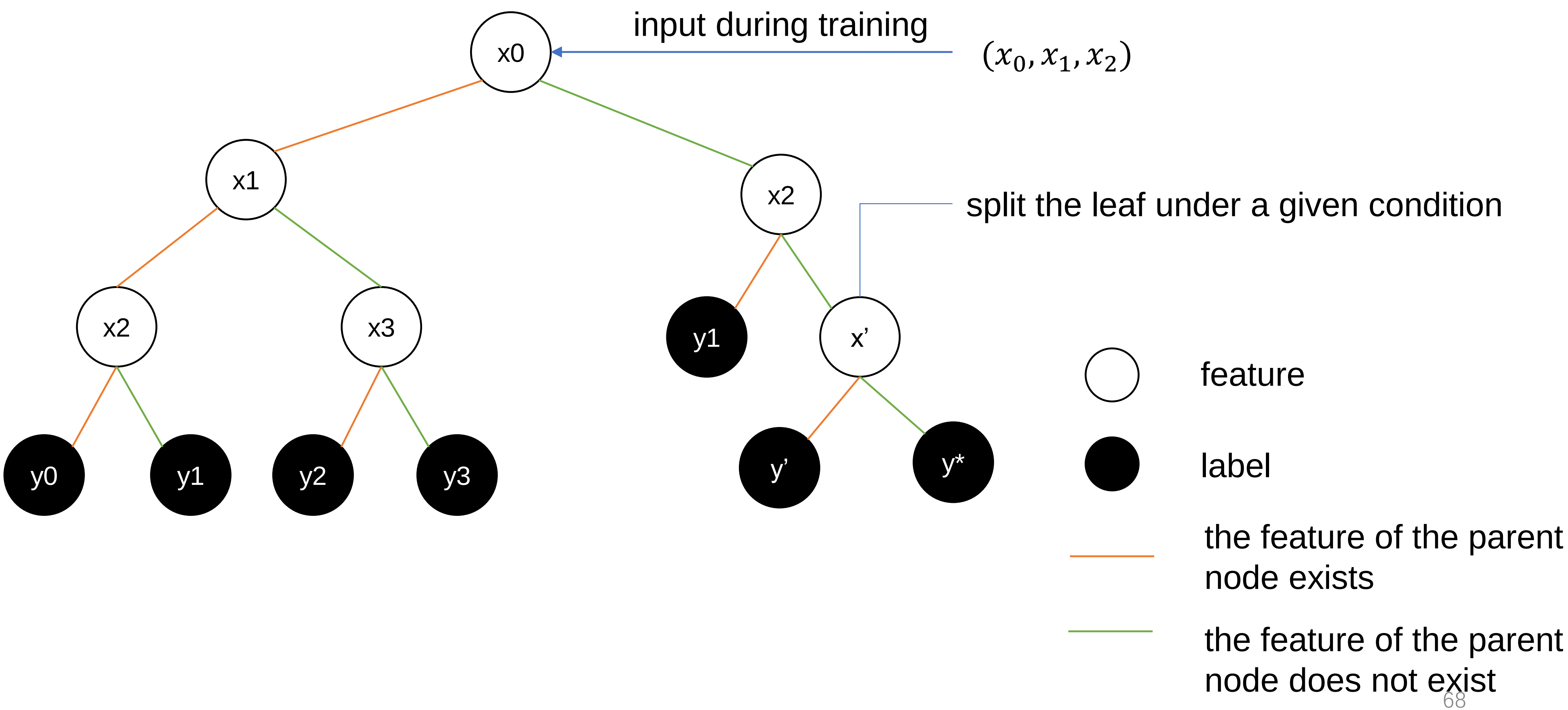
```
Theorem plus_O_n : forall n:nat, 0 + n = n.
Proof.
  simpl.
  reflexivity.
Qed.

Theorem mult_0_plus : forall n m : nat,
   (0 + n) * m = n * m.
Proof.
   intros n m.
   rewrite plus_O_n.
   reflexivity.
Qed.
```

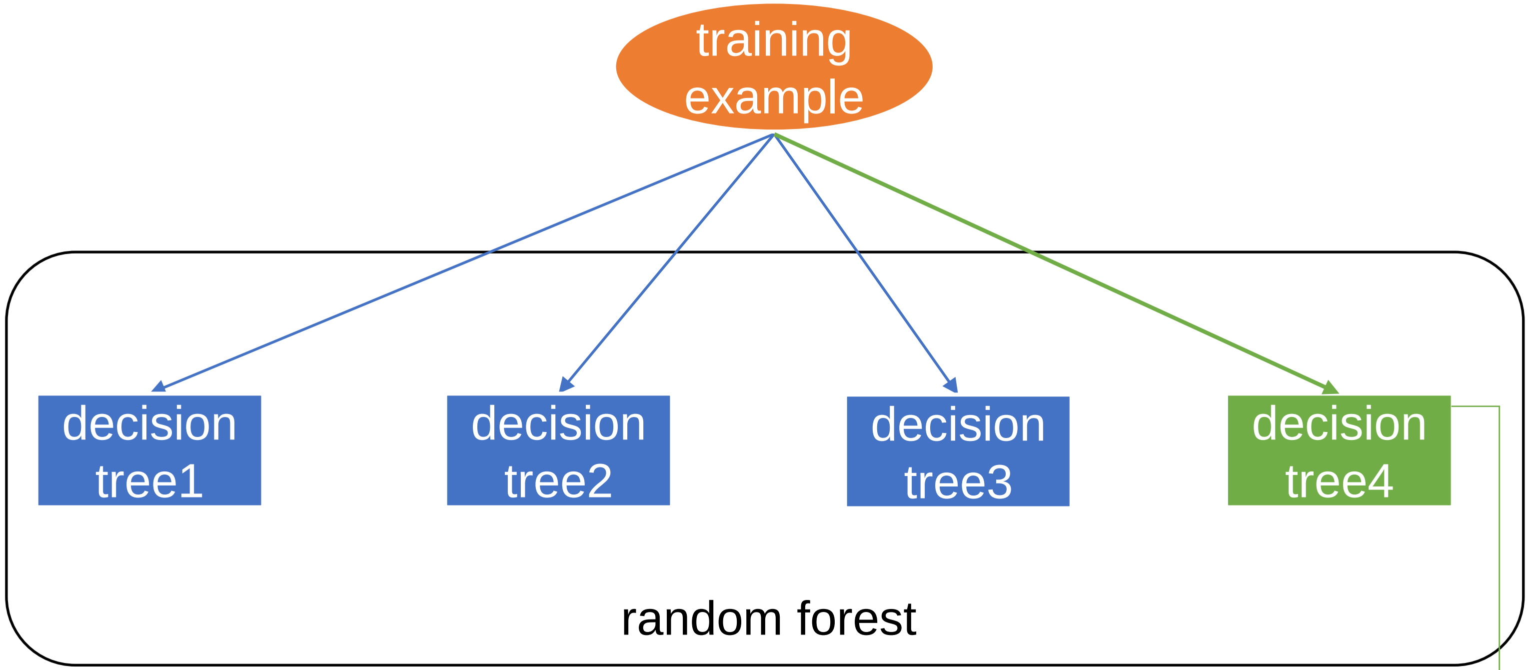
Online Random Forests (1/3)



Online Random Forests (2/3)



Online Random Forests (3/3)



- Make a new tree when an example is passed to the forest
  - with the probability of $\frac{1}{n}$
  - n is the number of trees in the forest
  - $\frac{1}{3}$ in the example