ATP-guidence for Learning Premise Selection

Bartosz Piotrowski, Josef Urban

29 III 2018
Aussois

Premise Selection

- Suppose we have:
  - a conjecture T we want to prove with Automated Theorem Prover (ATP),
  - a set of premises P (axioms, definitions, other theorems) we can use to prove T.
- If P is big, giving it all as axioms to the ATP for proving T is very likely to swamp the prover.
- So the task is to choose a *reasonably small* subset $P' \subseteq P$ such that still $P' \models T$.

In large formal libraries there is a lot of premises available. Therefore the task of Premise Selection is crucial to make proving with ATPs succesful in these libraries.

Mizar Mathematical Library

- A vast collection of mathematical theorems and its proofs formalized and verified in the Mizar system.
- In the current version of MML there is > 57000 theorems and lemmas organized into 1305 articles.
- The library begins with the article from 1989 which contains axioms of Tarski-Grothendieck set theory. In subsequent articles we find theorems from various classical areas of mathematics, for example:
  - Borsuk–Ulam theorem,
  - Jordan Curve theorem,
  - Brouwer Fixed Point theorem,
  - Hahn-Banach theorem,
  - Ramsey's theorem,
  - Gödel's Completeness theorem,
  - Schröder-Bernstein theorem,
  - ...

Mizar Mathematical Library – examples

```
theorem :: IRRAT_1:2
  ex x, y being real number st
  (x is irrational & y is irrational & x ^ y is rational)

theorem :: CARD_5:31
  for a, b being Aleph st a c= b holds
  exp(a,b) = exp(2,b)

theorem :: TOPS_3:27
  for X being non empty TopSpace
  for A being Subset of X holds
  (A is nowhere_dense iff ex C being Subset of X st
  (A c= C & C is closed & C is boundary))
```

In HTMLised MML: IRRAT_1:2, CARD_5:31, TOPS_3:27.

Some time ago Josef created the MPTP system for exporting statements of Mizar theorems, lemmas and definitions to the first-order TPTP language (which is standard for current first-order Automated Theorem Provers).

Mizar Mathematical Library in TPTP

Before:

```
theorem :: CARD_5:31
  for a, b being Aleph st a c= b holds
  exp(a,b) = exp(2,b)
```

After:

```
fof(t31_card_5, conjecture,
(![A]: ((~(v1_finset_1(A)) & v1_card_1(A)) =>
(![B]: ((~(v1_finset_1(B)) & v1_card_1(B)) =>
(r1_ordinal1(A,B) => k3_card_2(A,B) = k3_card_2(2,B))))))
```

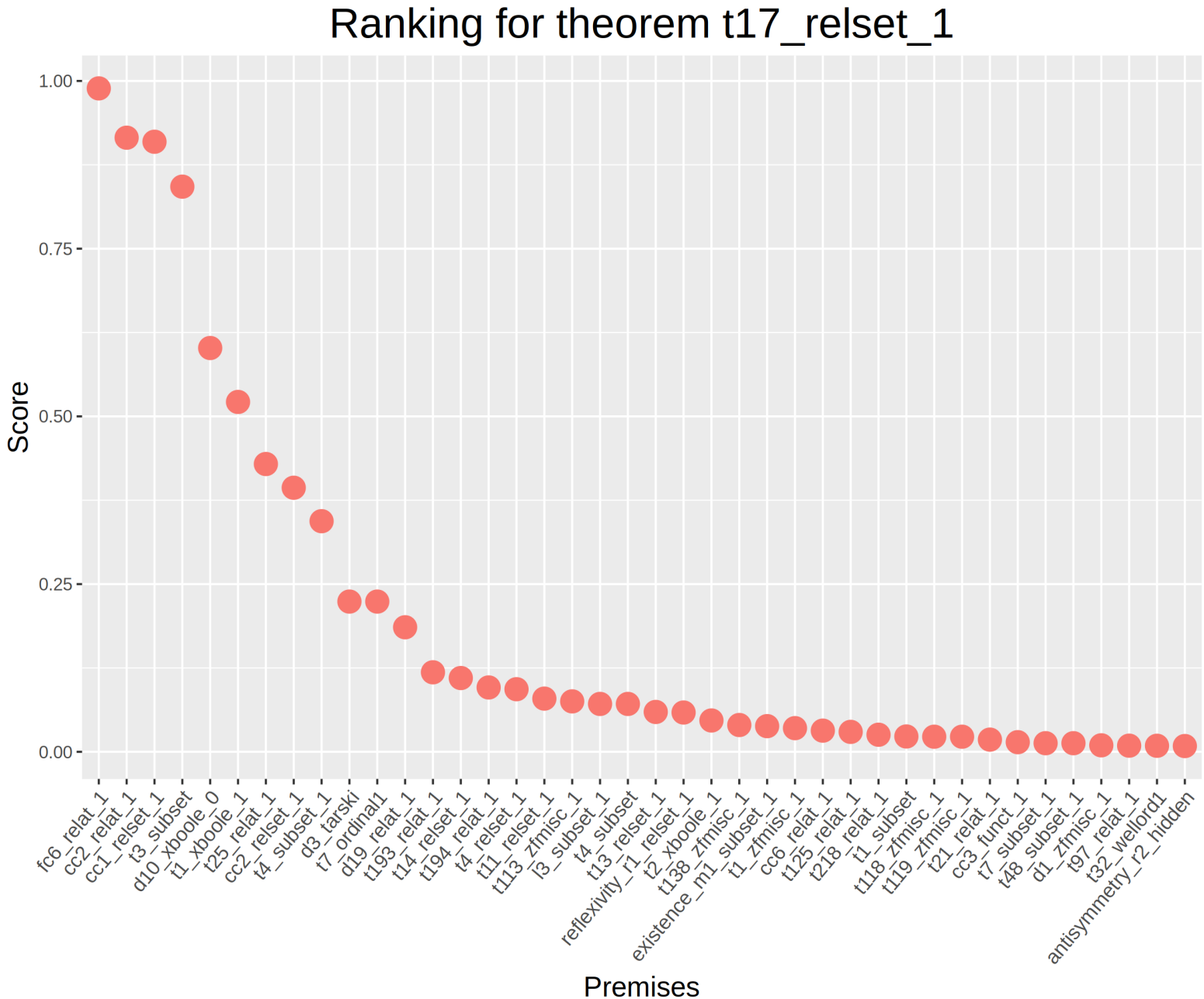
Mizar Mathematical Library in TPTP

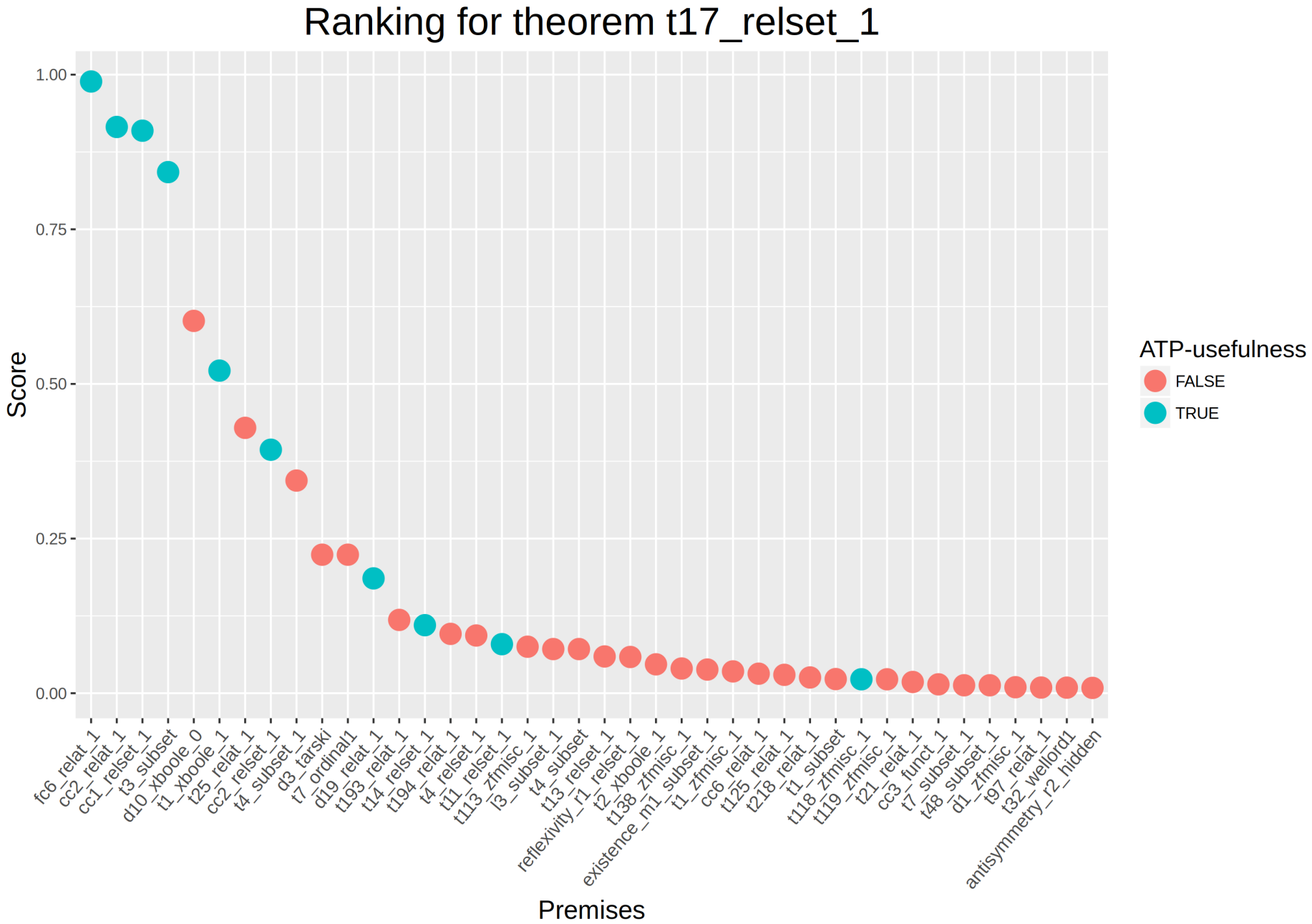
- In the corpus there is 57917 theorems, which along with definitions, axioms, typing statements and schema instantiations give 146700 premises.
- Currently, out of these 57917 theorems roughly 33000 can be proved automatically (with reasonably modest computational resources) having provided an adequate subset of premises as axioms and an appropriate set of parameters governing a strategy of a given ATP.
- In our experiments with Premise Selection we limit ourselves to these 33000 *ATP-provable* theorems.

Machine Learning for Premise Selection

Currently, most efficient methods for Premise Selection are based on *data-driven/machine-learning* approaches, where the notion of *useful premise* is learned from the known proofs.
The procedure is the following:

1. On a basis of a set of theorems with its proofs we create a set of training examples which is meant to sufficiently illustrate the notion of a *useful premise*,
2. We train a machine learning model on the prepared examples.
3. We create rankings of relevat premises for a separate test set of theorems, according to the model.
4. Finally, we evaluate the performance of the model by trying to prove the test theorems with ATP using premises from several top slices of the created rankings.


Ranking for theorem t17_relset_1
Score
1.00
0.75
0.50
0.25
0.00
Premises
fc6_relat_1
cc2_relat_1
cc1_relset_1
t3_subset
d10_xboole_0
t1_xboole_1
t25_relat_1
cc2_relset_1
t4_subset_1
d3_tarski
t7_ordinal1
d19_relat_1
t193_relat_1
t14_relset_1
t194_relat_1
t4_relset_1
t11_relset_1
t113_zfmisc_1
l3_subset_1
t4_subset
t13_relset_1
reflexivity_r1_relset_1
t2_xboole_1
t138_zfmisc_1
existence_m1_subset_1
t1_zfmisc_1
cc6_relat_1
t125_relat_1
t218_relat_1
t1_subset
t118_zfmisc_1
t119_zfmisc_1
t21_relat_1
cc3_funct_1
t7_subset_1
t48_subset_1
d1_zfmisc_1
t97_relat_1
t32_wellord1
antisymmetry_r2_hidden


Ranking for theorem t17_relset_1
Score
1.00
0.75
0.50
0.25
0.00
Premises
fc6_relat_1
cc2_relat_1
cc1_relset_1
t3_subset
d10_xboole_0
t1_xboole_1
t25_relat_1
cc2_relset_1
t4_subset_1
d3_tarski
t7_ordinal1
d19_relat_1
t193_relat_1
t14_relset_1
t194_relat_1
t4_relset_1
t11_relset_1
t113_zfmisc_1
l3_subset_1
t4_subset
t13_relset_1
reflexivity_r1_relset_1
t2_xboole_1
t138_zfmisc_1
existence_m1_subset_1
t1_zfmisc_1
cc6_relat_1
t125_relat_1
t218_relat_1
t1_subset
t118_zfmisc_1
t119_zfmisc_1
t21_relat_1
cc3_funct_1
t7_subset_1
t48_subset_1
d1_zfmisc_1
t97_relat_1
t32_wellord1
antisymmetry_r2_hidden
ATP-usefulness
FALSE
TRUE

There are several specific peculiarities of applying Machine Learning to Premise Selection, which makes this problem interesting and challenging.

How to represent mathematical formulae?

ML methods typically need good feature representation of objects they operate on; it is highly non obvious how to characterize mathematical formulas that the representation is simple enough to be appropriate for ML algorithm but also meaningful enough. Possible approaches are:

- syntactic features,
- model-based features,
- deep neural embeddings,
- ...

An example of a feature representation

Theorem CARD_5:31 in TPTP format:

```
fof(t31_card_5, conjecture,
(![A]: ((~(v1_finset_1(A)) & v1_card_1(A)) =>
(![B]: ((~(v1_finset_1(B)) & v1_card_1(B)) =>
(r1_ordinal1(A,B) => k3_card_2(A,B) = k3_card_2(2,B))))))).
```

... and its feature description:

```
"=", "2", "v1_card_1", "r1_ordinal1",
"v1_finset_1", "k3_card_2",
"=(k3_card_2(V,V), k3_card_2(2,V))", "=(V,V)",
"v1_card_1(V)", "v1_finset_1(V)", "r1_ordinal1(V,V)",
"k3_card_2(2,V)", "k3_card_2(V,V)",
"v1_card_1-V", "r1_ordinal1-V", "v1_finset_1-V",
"=-k3_card_2", "k3_card_2-V", "k3_card_2-2",
```

An example of feature representation

Such a set of features ...

```
"=", "2", "v1_card_1", "r1_ordinal1",
"v1_finset_1", "k3_card_2",
"=(k3_card_2(V,V), k3_card_2(2,V))", "=(V,V)",
"v1_card_1(V)", "v1_finset_1(V)", "r1_ordinal1(V,V)",
"k3_card_2(2,V)", "k3_card_2(V,V)",
"v1_card_1-V", "r1_ordinal1-V", "v1_finset_1-V",
"=-k3_card_2", "k3_card_2-V", "k3_card_2-2",
```

... is represented as a long, sparse binary vector:

```
(0, 1, 1, 0, 0, 0, 1, ..., 0, 0, 1, 0, 0)
```

Because featuresing the whole MML produces 451706 different features, length of such vector is 451706.

Which type of Machine Learning models?

There are two possible settings in which we can approach premise selection with Machine Learning:

1. *Multilabel setting*: here we treat premises used in the proofs as opaque labels on theorems and we train a model capable of labeling conjectures based on their features. Example:

   `[(0, 1, 0, 0, ..., 0), (prm_1, prm_2, prm_4)]`

2. *Binary setting*: here the aim of the learning model is to recognize pairwise-relevance of the *conjecture-premise* pairs, i.e. to decide what is the chance of the premise being relevant for proving the conjecture based on the features of both the conjecture and the premise. Example:

   `[(0, 1, 0, 0, ..., 0), (1, 0, 0, 0, ..., 1), 0]`

- Most of the work on premise selection was done in *multilabel setting* with use of fast and simple ML algorithms like k-NN and Naive Bayes.
- In 2016/17, Google Research team led by Christian Szegedy applied deep neural networks in the *binary setting* and improved state-of-the-art in the field. In deep learning approach neural net learns feature representation on its own.
- Our goal was to do premise selection in binary setting with some ML algorithm which is easier to train, with handcrafted features.
- We have chosen XGBoost as ML algorithm to use, because it proved be efficient in many ML competitions, it is fast, and performs well with sparse features.

How to create a good training set?

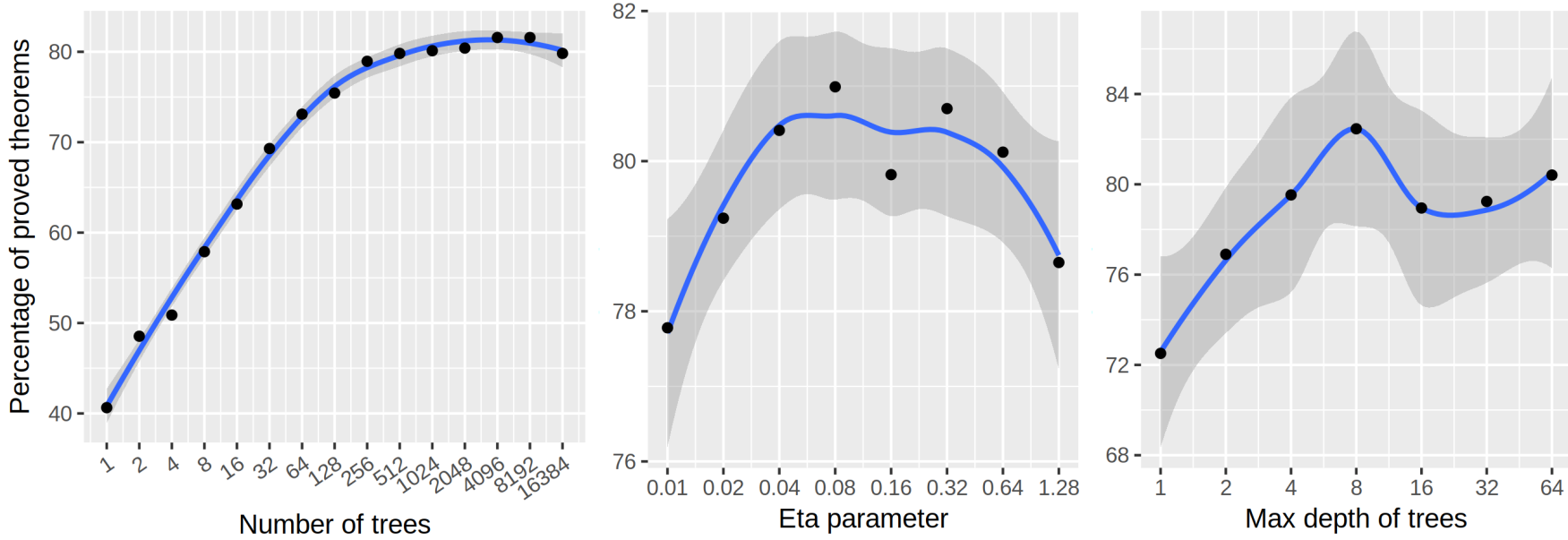
- Theorems have many proofs with various sets of premises used in them. It would be very difficult to generate all ATP-feasible proofs for some set of theorems.
- Different proofs have different *difficulty* for the given ATP.
- Different provers find different proofs.

It means it is tricky to prepare an appropriate training set, because our environment is to some extent unknown.
This is especially significant if we need to have also negative examples in the training set: if the theorem T can be proved with premises $\{a, b\}$ and $\{a, c\}$ but we, not being aware of the latter proof, present c as a negative example, we can distort the process of learning the notion of useful premise.

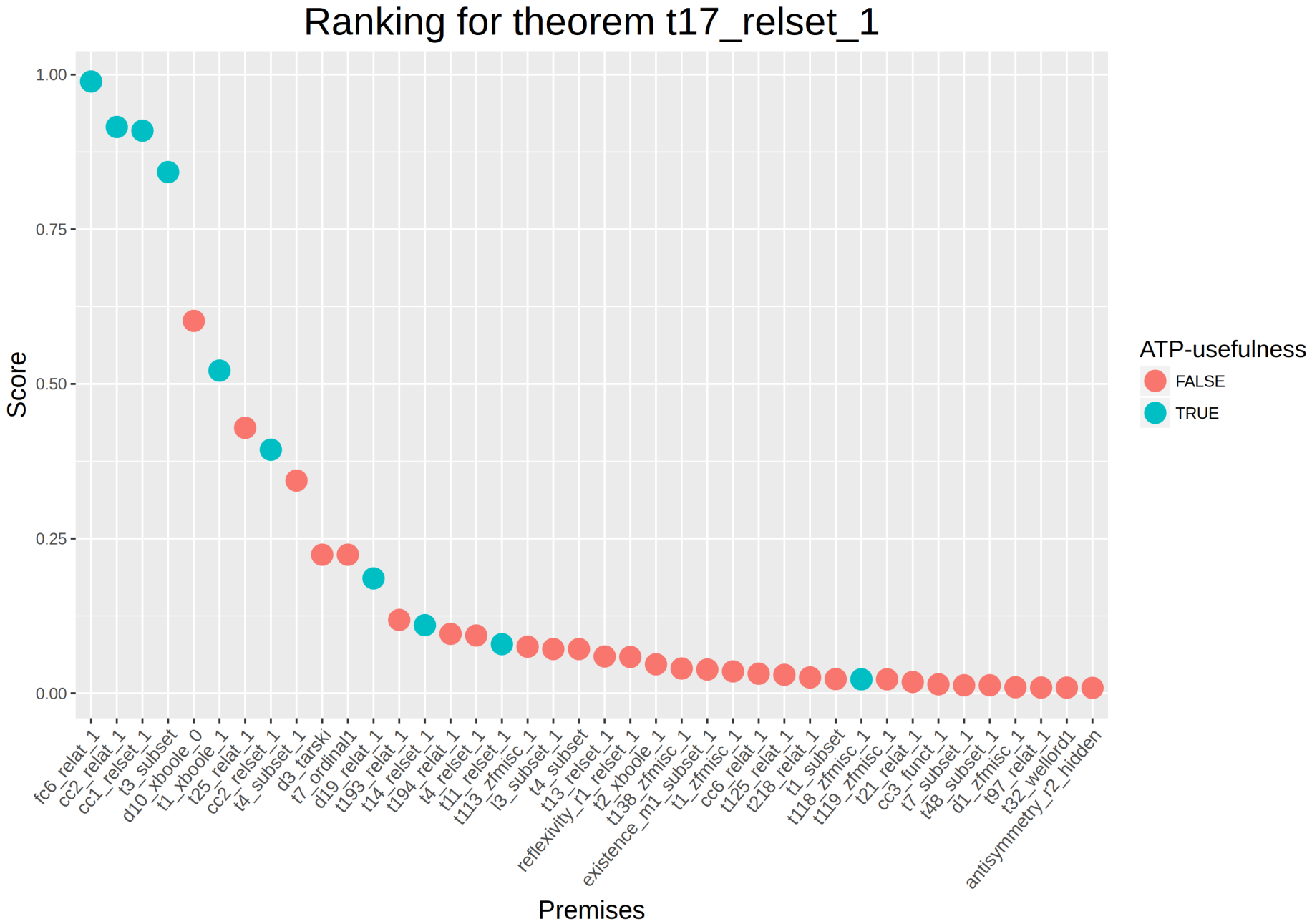
How to evaluate performance of the learned model?

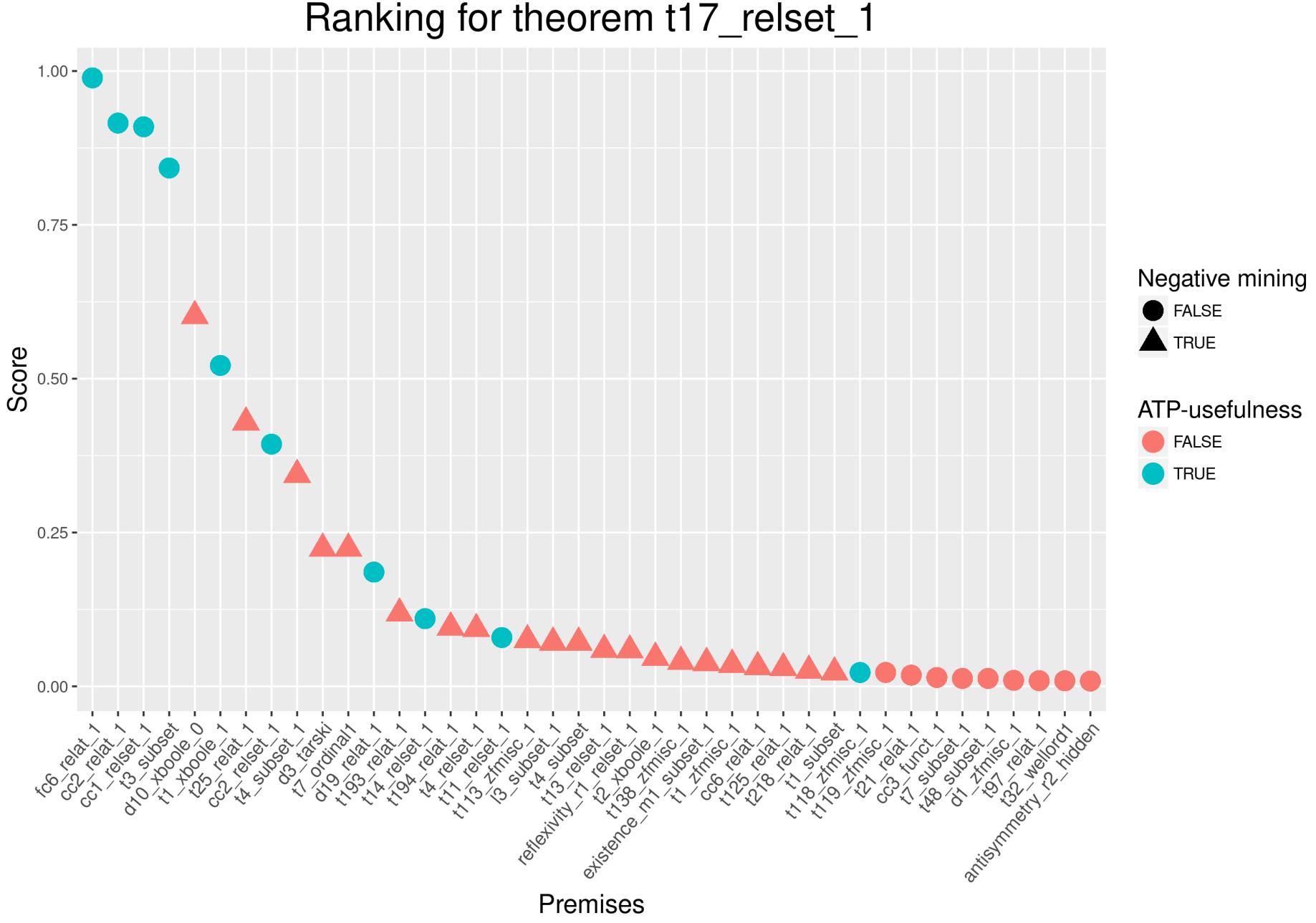
- Doing evaluation we cannot fully relay on the already known proofs of the theorems from the test set, because the Machine Learning model often can surprise us with proposing such premises that they result in a new, so far unknown proof – so ATP-evaluation is much more meaningful.
- New proofs found ATP-evaluation can be useful for subsequent round of training.
- It is beneficial to do ATP-minimization of proofs.
- Because ATPs have many powerful heuristics, it is good to intentionally limit their capabilities when comparing different Premise Selection methods, in order to isolate effects of these methods. We use E prover with `auto` mode and we put at most 512 premises as axioms for proving.

To tune parameters of XGBoost model, training/test split was fixed and each trained model was evaluated with ATP:



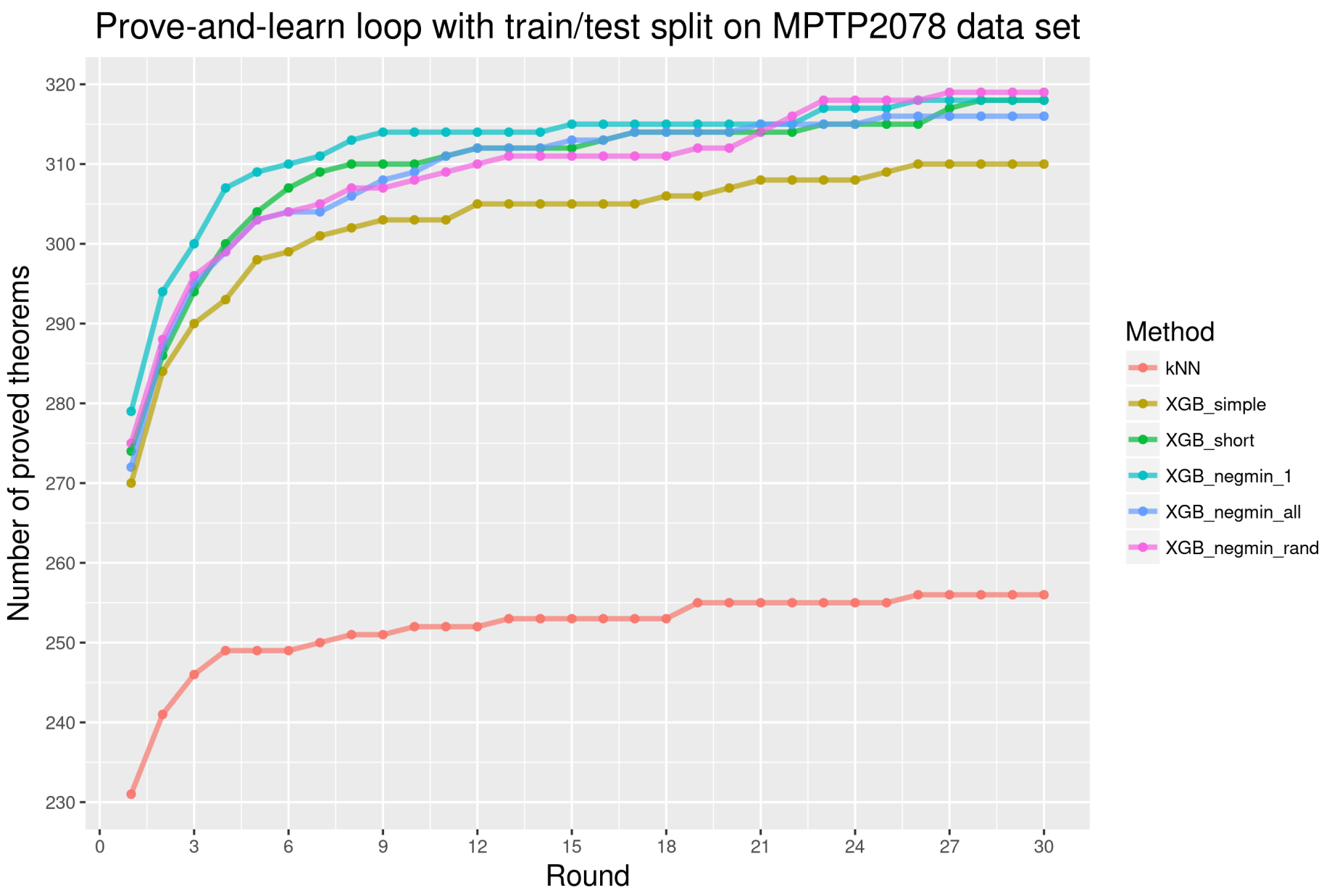
Observations from previous slides suggest to introduce a setup of experiments in which there is an interaction between a given ATP and a Machine Learning algorithm.
In the spirit of these remarks we started loop-style experiments where we intersperse training and ATP-evaluation, applying *ATP-negative mining* to create a training set.


Ranking for theorem t17_relset_1
Score
1.00
0.75
0.50
0.25
0.00
Premises
fc6_relat_1
cc2_relat_1
cc1_relset_1
t3_subset
d10_xboole_0
t1_xboole_1
t25_relat_1
cc2_relset_1
t4_subset_1
d3_tarski
t7_ordinal1
d19_relat_1
t193_relat_1
t14_relset_1
t194_relat_1
t4_relset_1
t11_relset_1
t113_zfmisc_1
l3_subset_1
t4_subset
t13_relset_1
reflexivity_r1_relset_1
t2_xboole_1
t138_zfmisc_1
existence_m1_subset_1
t1_zfmisc_1
cc6_relat_1
t125_relat_1
t218_relat_1
t1_subset
t118_zfmisc_1
t119_zfmisc_1
t21_relat_1
cc3_funct_1
t7_subset_1
t48_subset_1
d1_zfmisc_1
t97_relat_1
t32_wellord1
antisymmetry_r2_hidden
ATP-usefulness
FALSE
TRUE


Ranking for theorem t17_relset_1
Score
1.00
0.75
0.50
0.25
0.00
Premises
fc6_relat_1
cc2_relat_1
cc1_relset_1
t3_subset
d10_xboole_0
t1_xboole_1
t25_relat_1
cc2_relset_1
t4_subset_1
d3_tarski
t7_ordinal1
d19_relat_1
t193_relat_1
t14_relset_1
t194_relat_1
t4_relset_1
t11_relset_1
t113_zfmisc_1
i3_subset_1
t4_subset
t13_relset_1
reflexivity_r1_relset_1
t2_xboole_1
t138_zfmisc_1
existence_m1_subset_1
t1_zfmisc_1
cc6_relat_1
t125_relat_1
t218_relat_1
t1_subset
t118_zfmisc_1
t119_zfmisc_1
t21_relat_1
cc3_funct_1
t7_subset_1
t48_subset_1
d1_zfmisc_1
t97_relat_1
t32_wellord1
antisymmetry_r2_hidden
Negative mining
FALSE
TRUE
ATP-usefulness
FALSE
TRUE

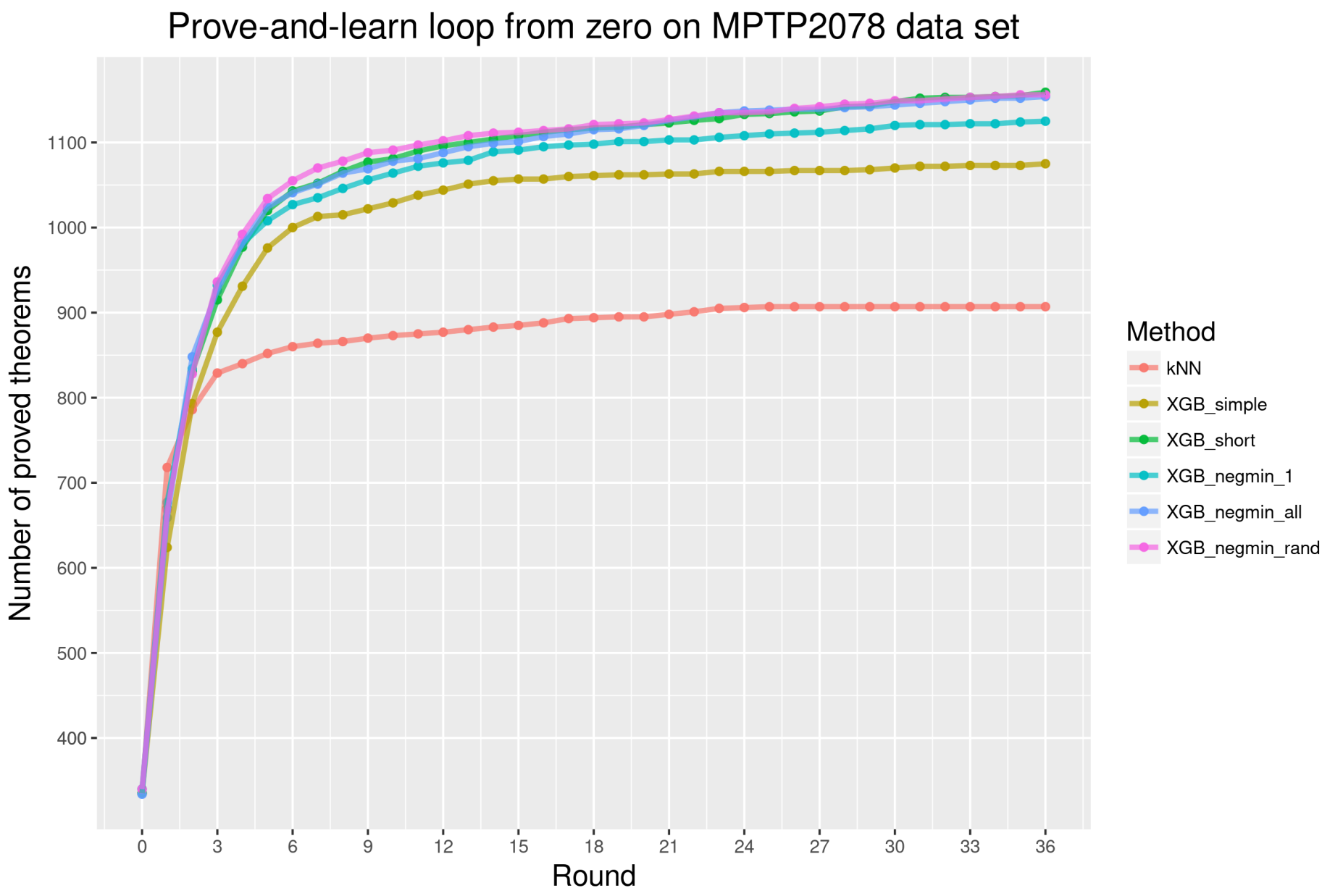
Prove-and-learn loop on train/test split

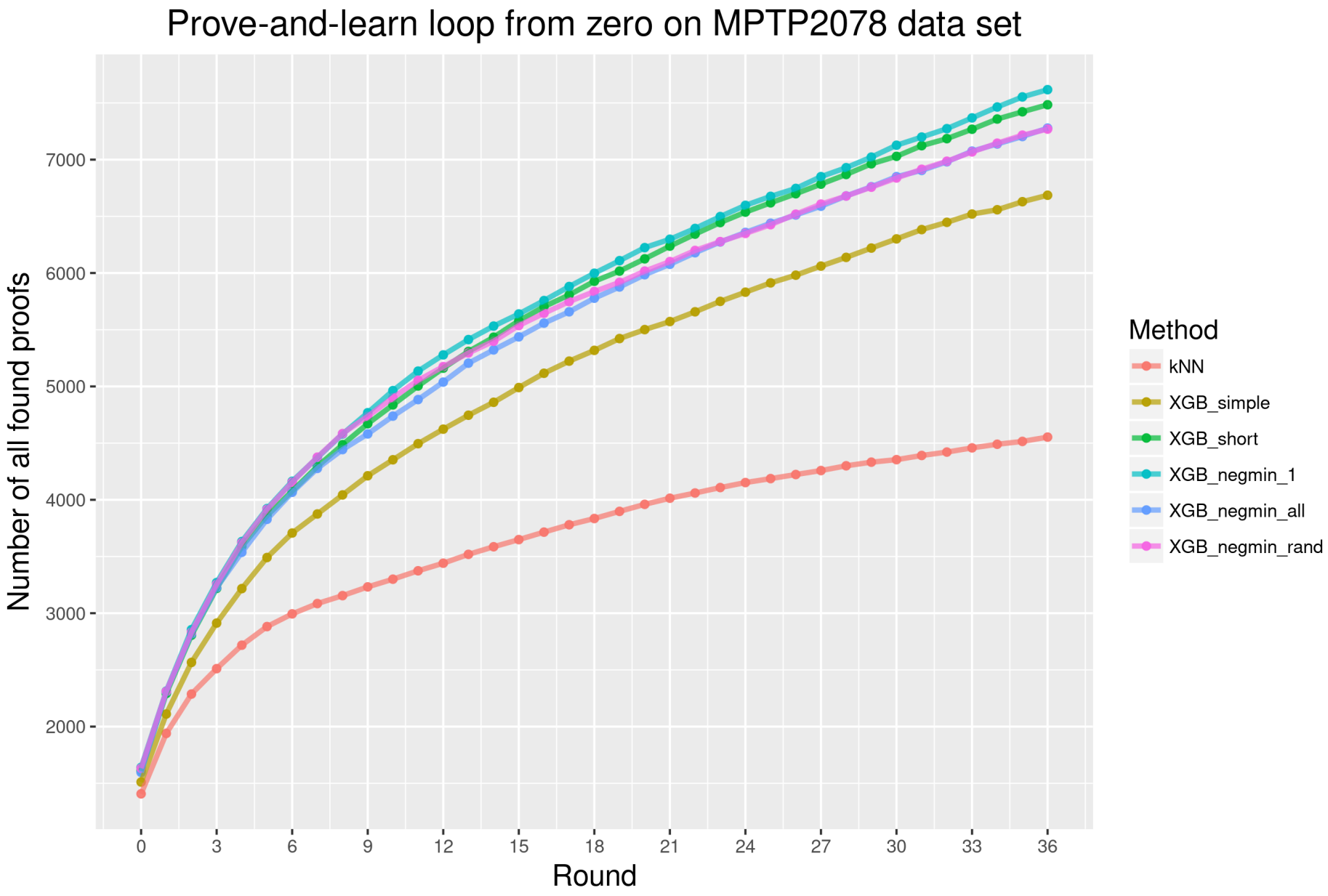
```
Require: theorems_train, proofs_train, theorems_test, premises

train_set = prepare_train_set(proofs_train)
while(more proofs for training theorems found)
{
  model = train(train_set)
  rankings_test = rankings(model, theorems_test, premises)
  proofs_test = atp_eval(rankings_test)
  rankings_train = rankings(model, theorems_train, premises)
  proofs_train = atp_eval(rankings_train)
  train_set = negative_mining(rankings_train, proofs_train)
}
```

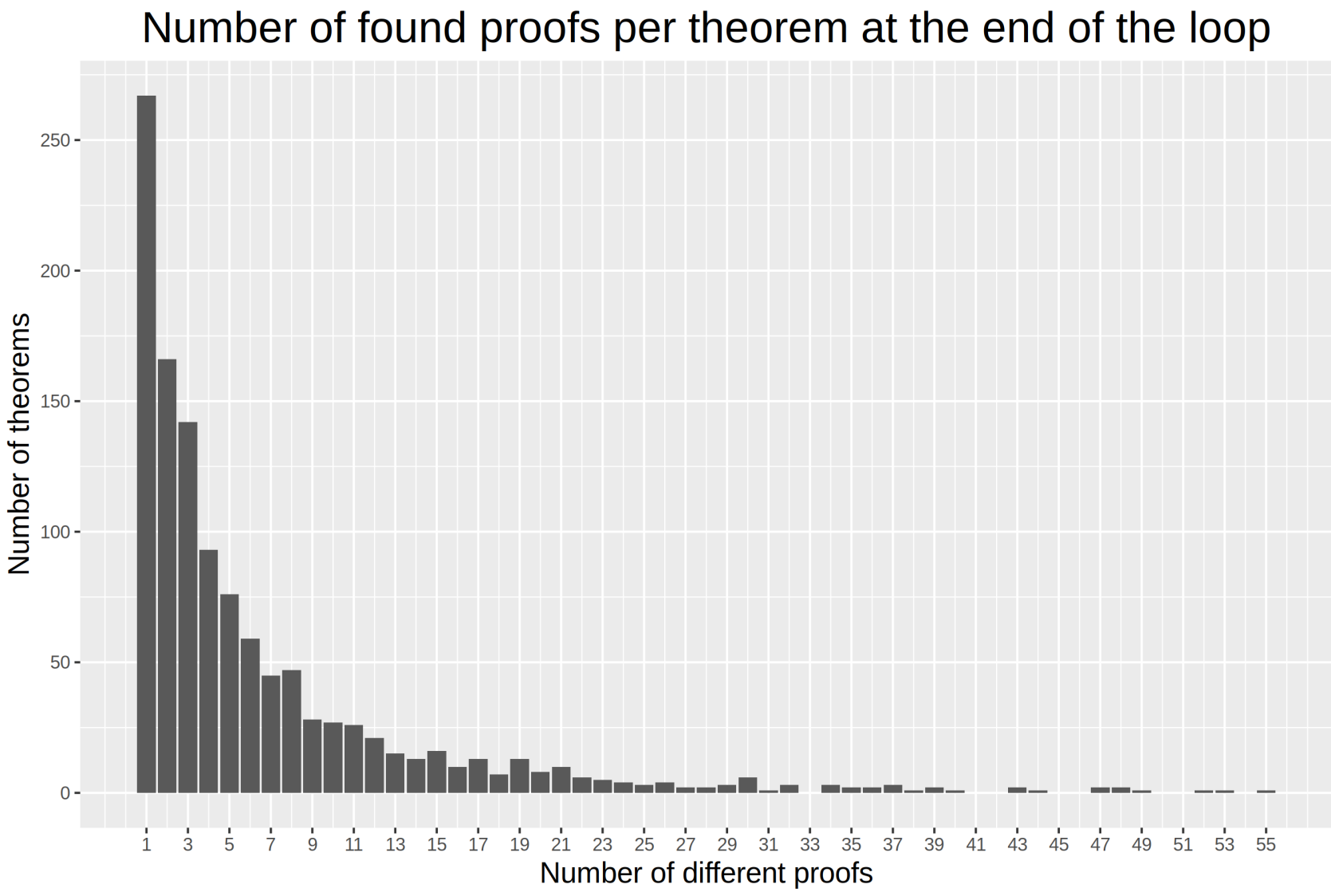

Prove-and-learn loop with train/test split on MPTP2078 data set
Number of proved theorems
320
310
300
290
280
270
260
250
240
230
0
3
6
9
12
15
18
21
24
27
30
Round
Method
kNN
XGB_simple
XGB_short
XGB_negmin_1
XGB_negmin_all
XGB_negmin_rand

Prove-and-learn loop from zero

```
Require: theorems, premises

rankings = random_rankings(theorems)
proofs = atp_eval(rankings)
train_set = prepare_train_set(proofs)
while(more proofs found)
{
  model = train(train_set)
  rankings = rankings(model, theorems, premises)
  proofs = atp_eval(rankings)
  train_set = negative_mining(rankings, proofs)
}
```


Prove-and-learn loop from zero on MPTP2078 data set
Number of proved theorems
1100
1000
900
800
700
600
500
400
0
3
6
9
12
15
18
21
24
27
30
33
36
Round
Method
kNN
XGB_simple
XGB_short
XGB_negmin_1
XGB_negmin_all
XGB_negmin_rand


Prove-and-learn loop from zero on MPTP2078 data set
Number of all found proofs
2000
3000
4000
5000
6000
7000
0
3
6
9
12
15
18
21
24
27
30
33
36
Round
Method
kNN
XGB_simple
XGB_short
XGB_negmin_1
XGB_negmin_all
XGB_negmin_rand


Number of found proofs per theorem at the end of the loop
Number of theorems
250
200
150
100
50
0
1
3
5
7
9
11
13
15
17
19
21
23
25
27
29
31
33
35
37
39
41
43
45
47
49
51
53
55
Number of different proofs

Implementation:
`https://github.com/BartoszPiotrowski/ATPboost`

Future work: explore ATP-negative mining with deep neural nets.

Thank you!