Silas: Dependable and High Performance Machine Learning

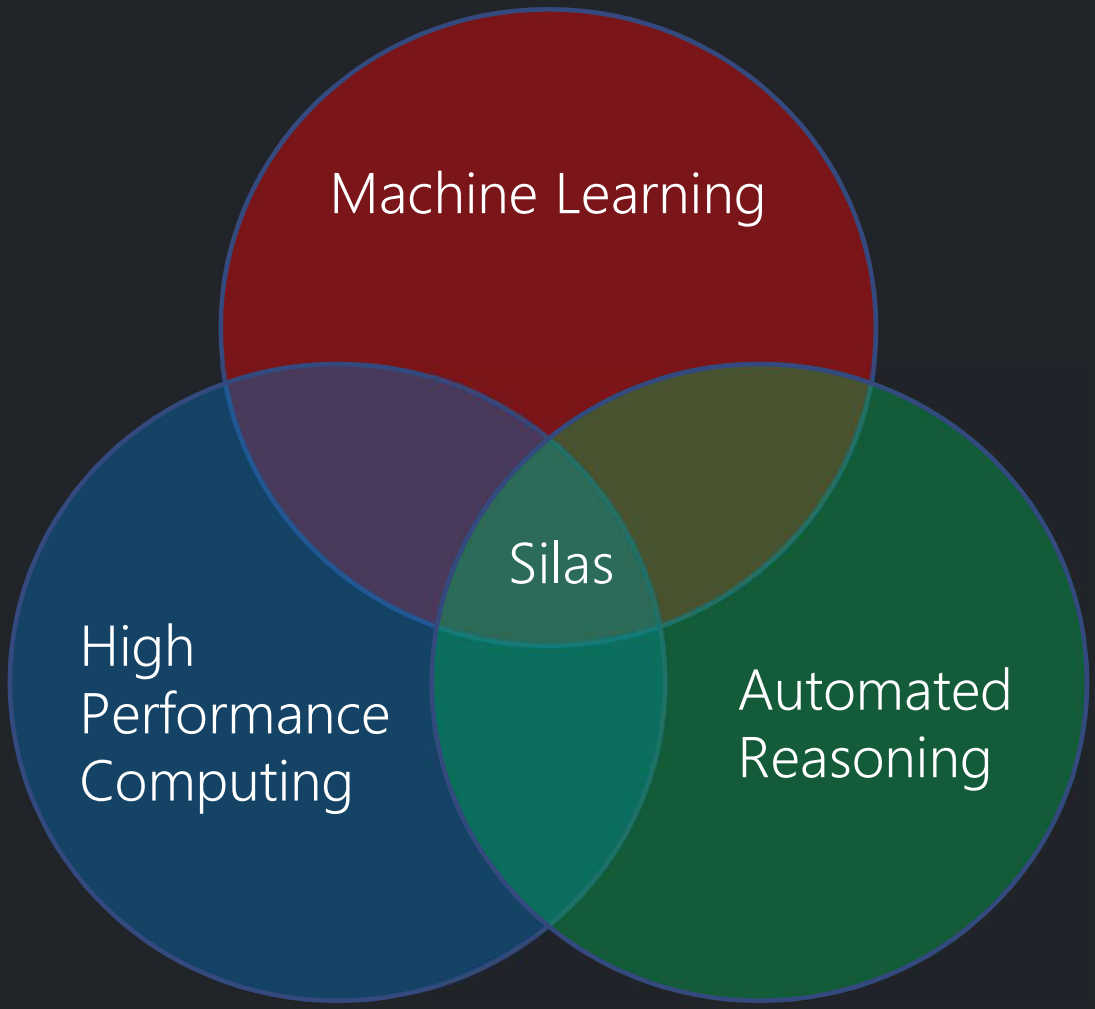
Prof. Jin Song Dong (Griffith)
Dr. Hadrien Bride (Griffith)
Dr. Zhe Hou (Griffith)

DEPINTEL

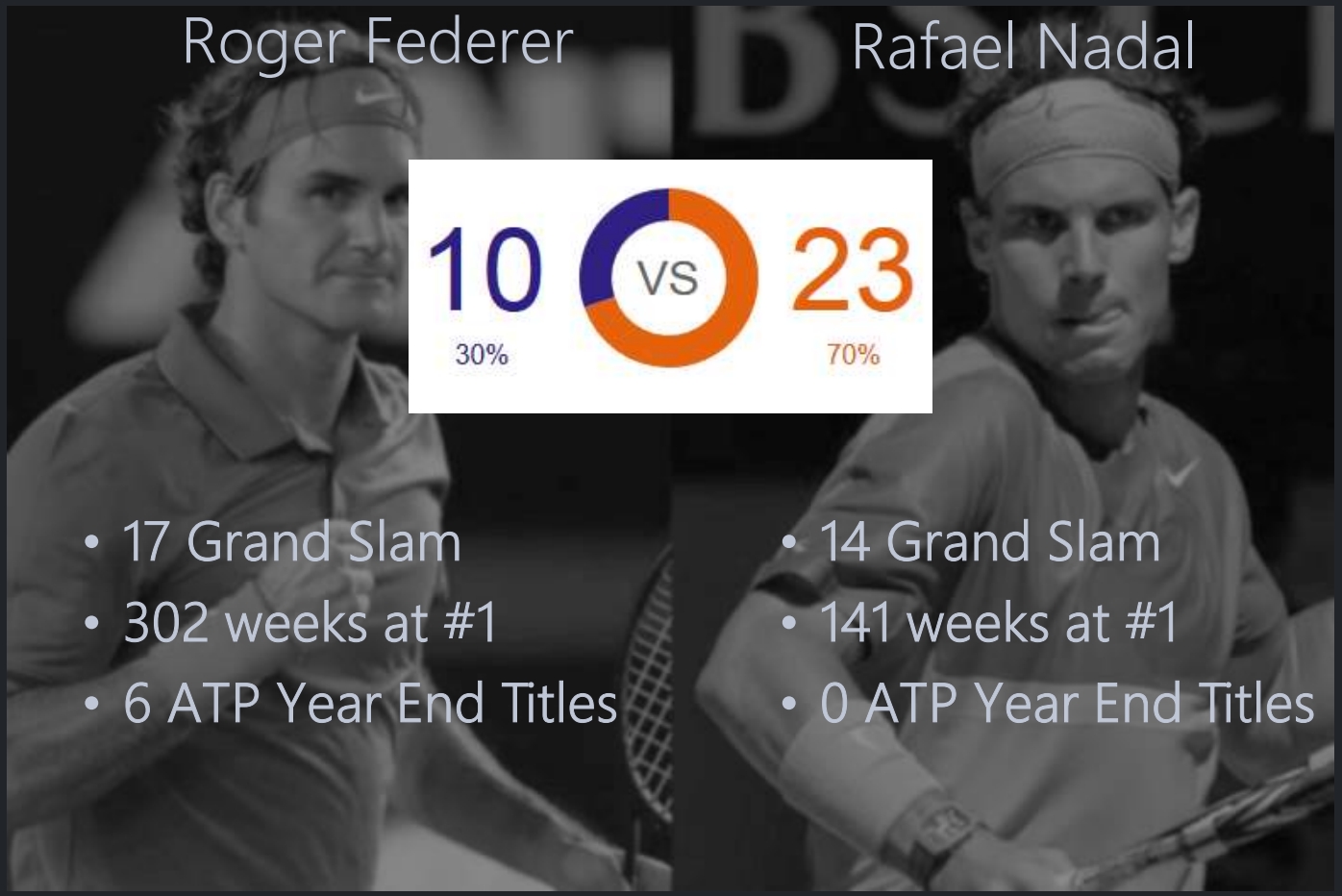
Dependable Intelligence

www.depintel.com

Silas


Machine Learning
Silas
High Performance Computing
Automated Reasoning

From A Garage Project



Before 2017

- Who wins?
- Why?
  - Strength/weakness?
  - Which part of the game is more important?
  - How to beat the opponent?
  - …

Model Checking with PAT

```
    Federer
 de_ct   ad_ct
 ------+------  baseline
 |  1  |  2  |
 |-----|-----|  service line
 |  3  |  4  |
 |===========|  net
 |  5  |  6  |
 |-----|-----|  service line
 |  7  |  8  |
 ------+------  baseline
 ad_ct   de_ct
    Nadal
```

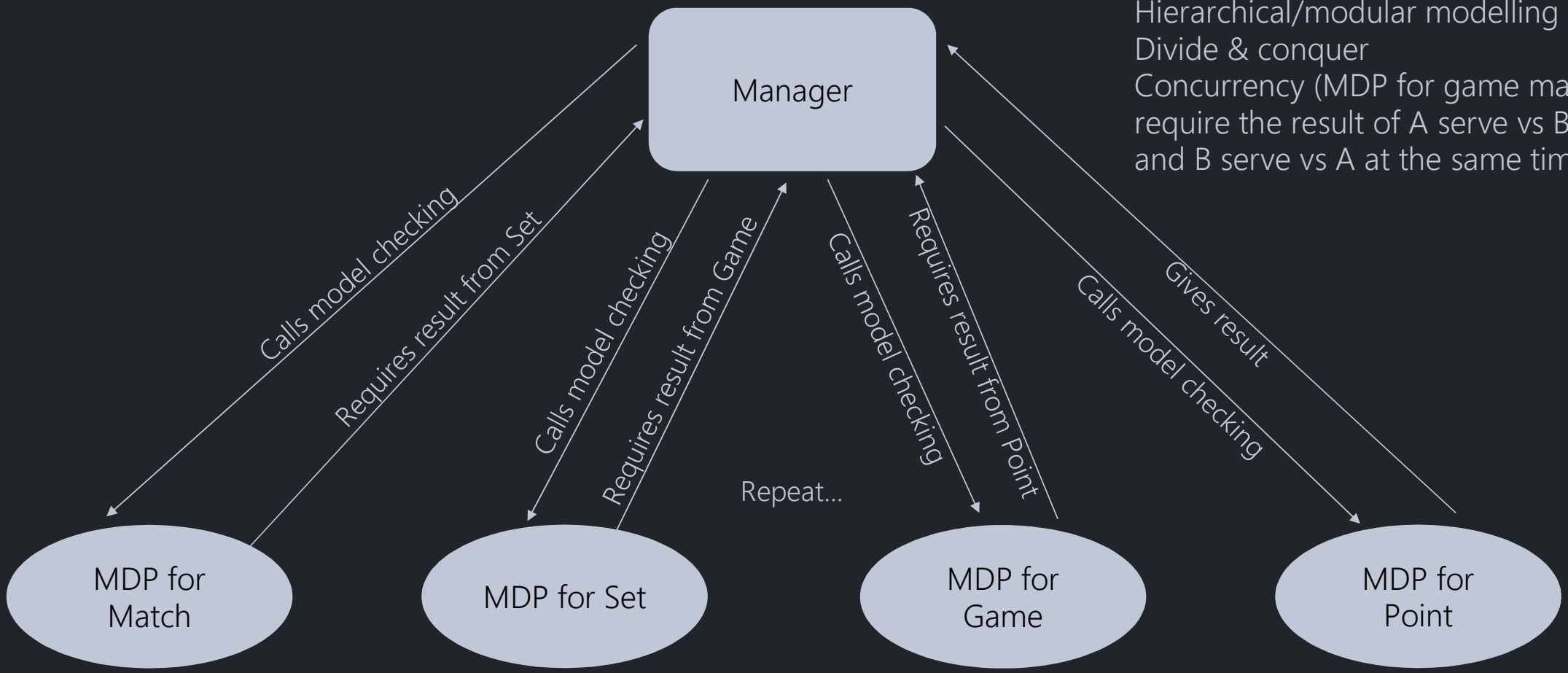
"9" represents net error or hit outside

MDP model of tie-break

PAT

The winner

Reflective Model Checking with PAT



Hierarchical/modular modelling
Divide & conquer
Concurrency (MDP for game may require the result of A serve vs B and B serve vs A at the same time)

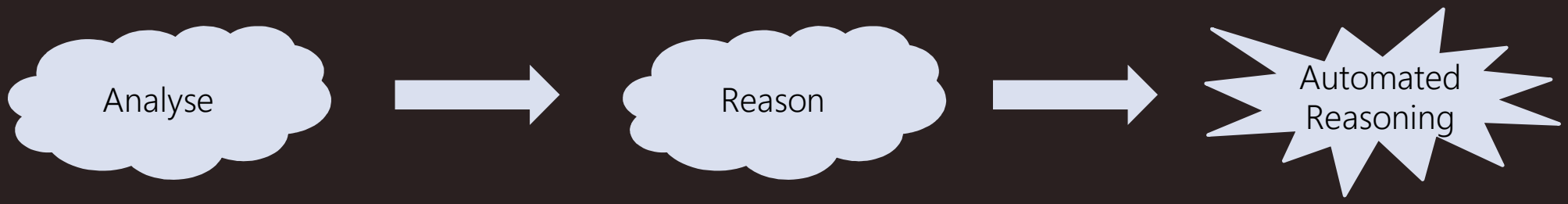
Predictive accuracy needs to be improved…

Machine Learning

- Not really live!
- Most machine learning techniques are still “black-boxes”.
  - Hard to explain what the model does.
  - Hard to interact with the model.
  - Hard to provide formal guarantee for the model.
- eXplainable AI (XAI) has been attracting attention.
- People have started looking into the interpretability of machine learning.
  - H2O, Lime etc. use linear regression to build approximation of the model and explain what the model roughly does.

- Who wins?
- Why?
  - Strength/weakness?
  - How to beat the opponent?
  - Which part of the game is more important?
  - …

Our Direction



There is plenty literature on applying machine learning in automated reasoning, such as Sledgehammer, but not the other way around.

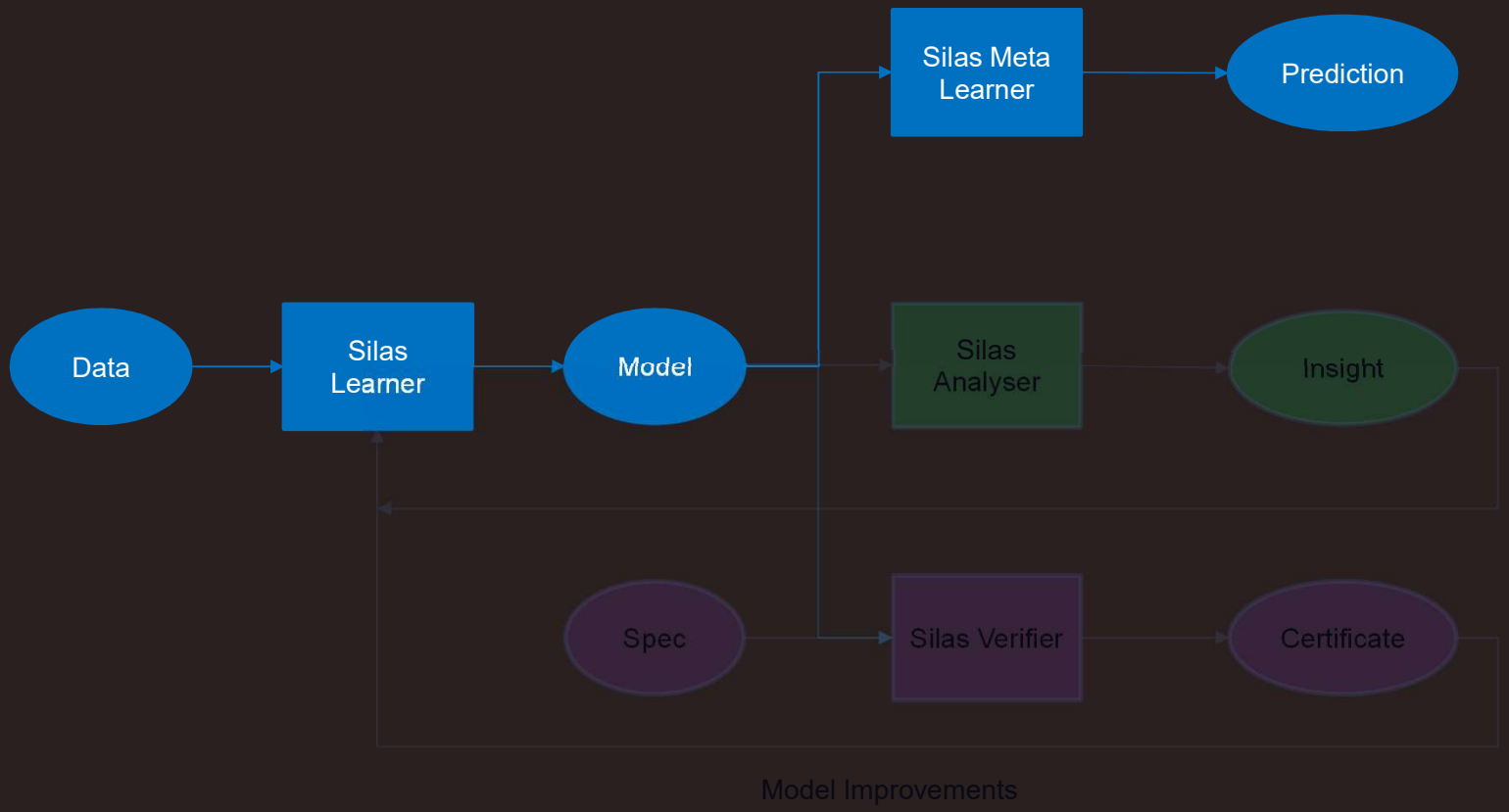
A Survey on Existing Machine Learning Techniques

Technique	Suitability for Logical Analysis	Predictive Performance	Time Efficiency	Space Efficiency
Linear Models	Good	Bad	Very Good	Very Good
SVM	Bad	Average	Average	Good
Bayesian Networks	Average	Good	Average	Average
Ensemble Trees	Good	Good	Good	Average
Deep Learning	Bad	Very Good	Bad	Bad

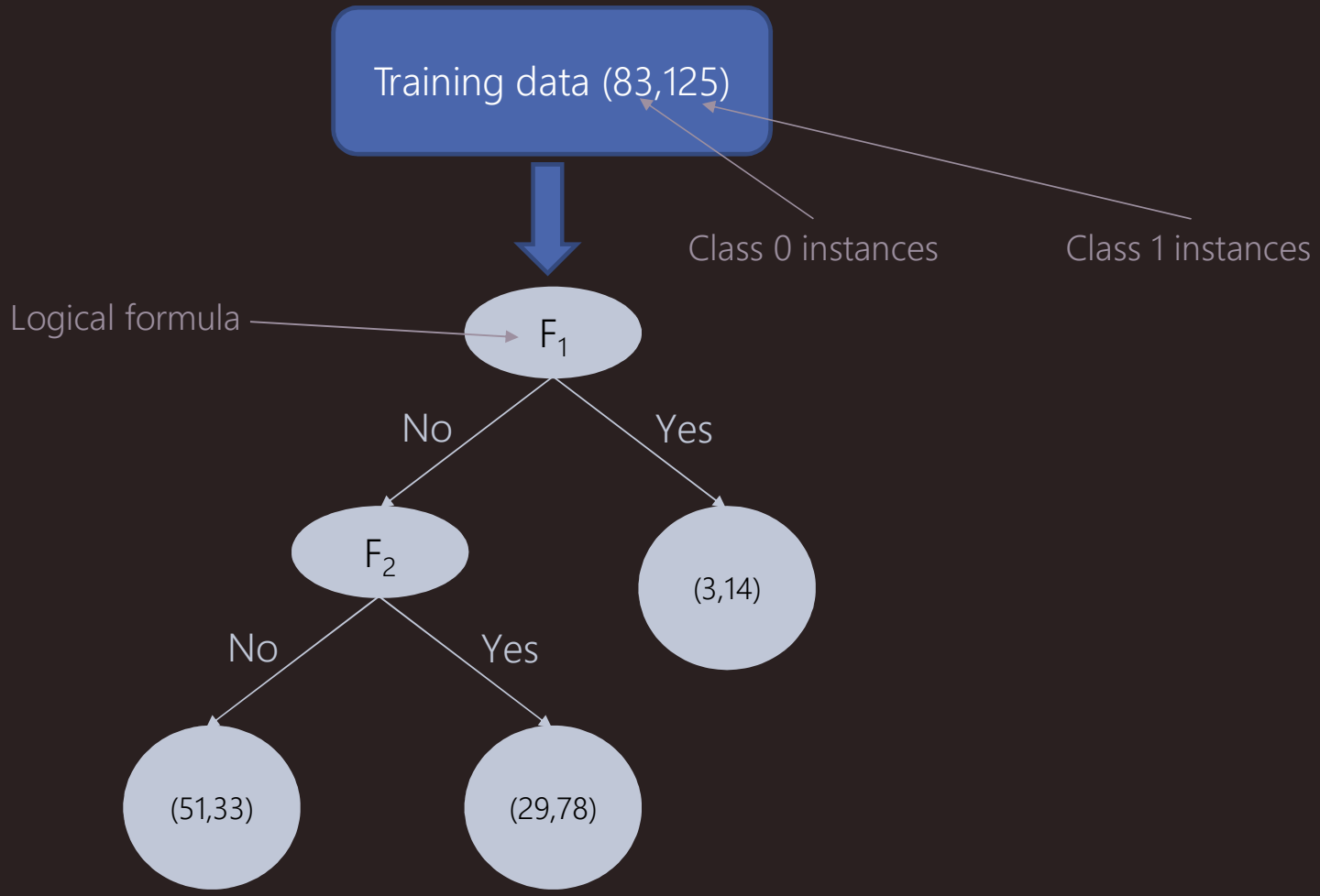
Ensemble Trees

- Based on decision trees.
  - Can be converted to formal semantics.
  - Widely-used in automated reasoning.
  - Easy to understand and modify.
- Excellent predictive performance on structured data.
- Computationally efficient.

Silas Machine Learning



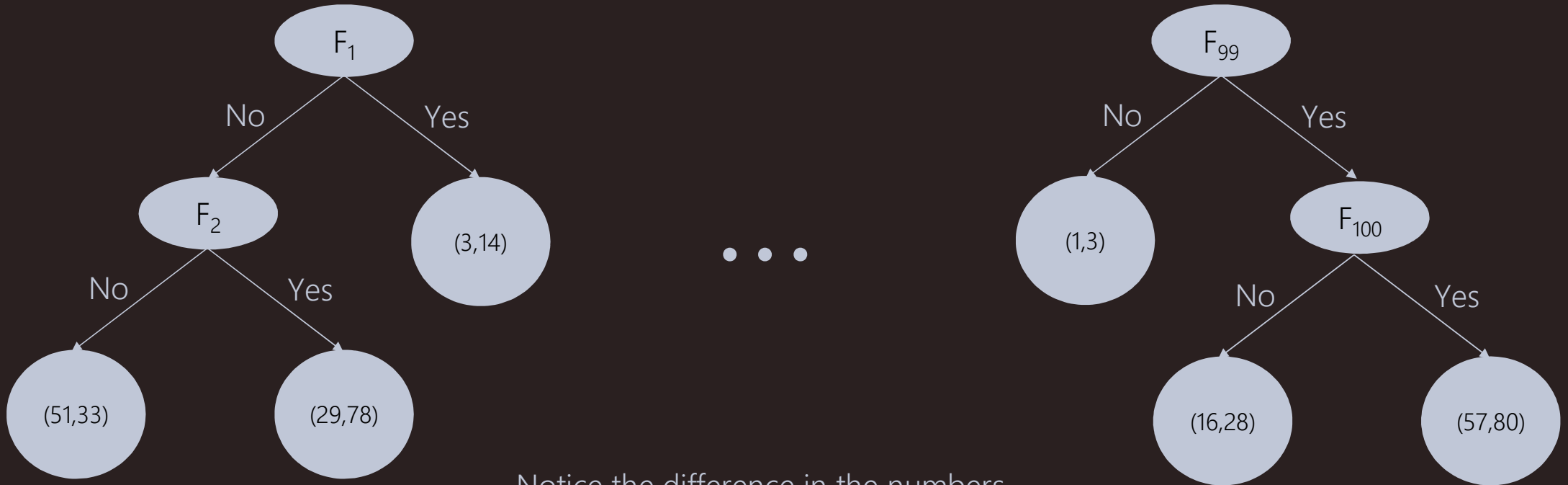
Decision Tree With a Logical Foundation



Decision Tree With a Logical Foundation

- The logical language of F is an extension of propositional logic.
  - Arithmetic term:
    - Atomic arithmetic term:
      - Variable (i.e., feature/attribute)
      - Constant (i.e., value)
    - Compound arithmetic term:
      - Unary operators: -, sqr
      - Nary operators: +, -, *, /
  - Boolean formula
    - Atomic Boolean formula
      - Boolean constant: true, false
      - Membership: variable $\in$ {constants}
      - Comparison: =, >, $\geq$, <, $\leq$ over arithmetic terms
    - Compound Boolean formula
      - Unary operator: $\neg$
      - Binary operators: $\wedge$, $\vee$, $\Rightarrow$, $\oplus$

Ensemble Trees

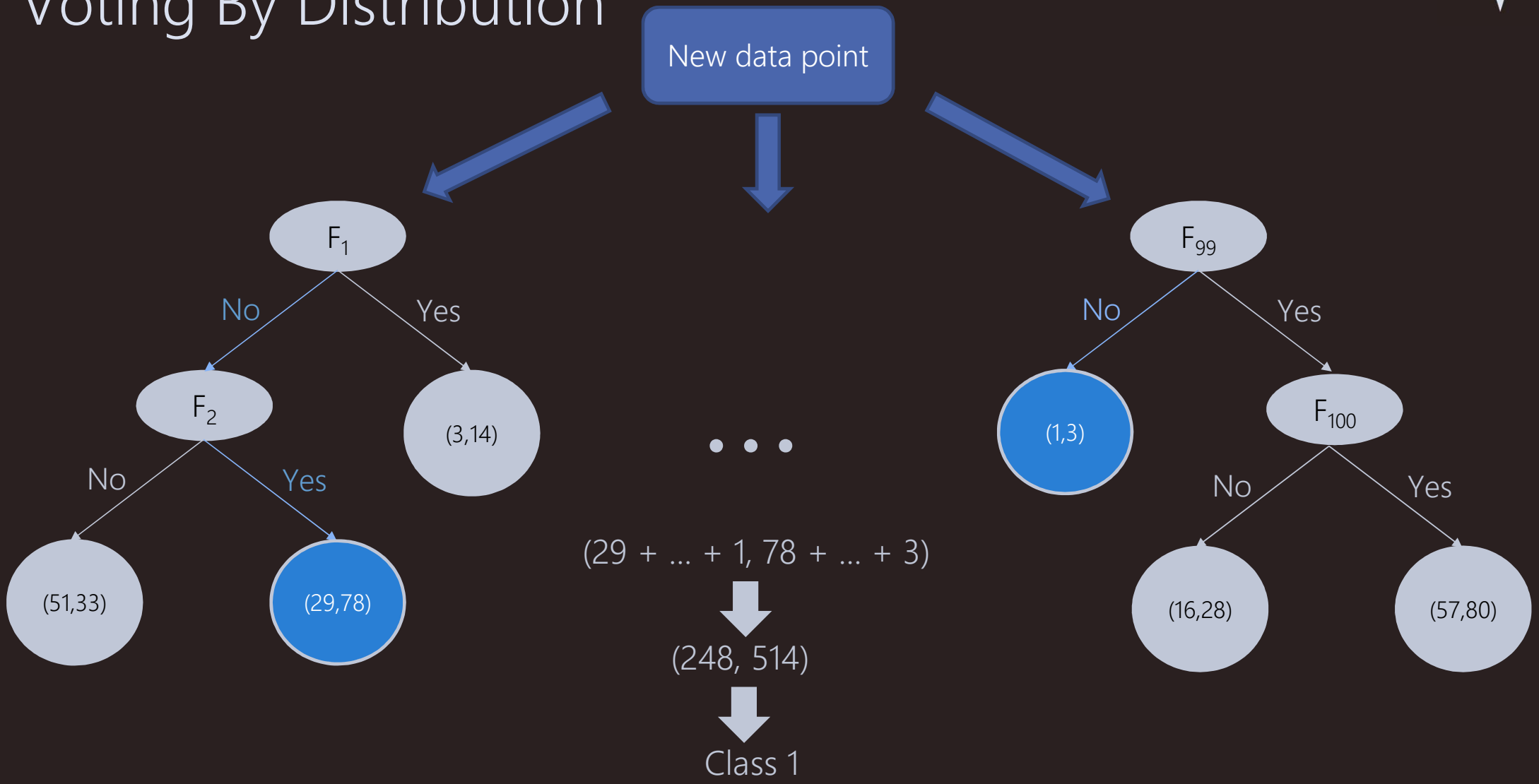


Notice the difference in the numbers…

The trees are trained using different samples via bagging.

Ensemble Trees

Voting By Distribution



Customised Algorithms

- Sampling
  - Sample training data for each tree.
  - Out-of-the-bag sampling.
  - Sample features for each tree.
- Decision formula searching
  - Numerical features
  - Enumeration features
  - Collection features

If you have ample time for computation…

Rank arithmetic/logical operators.

Rank features and collection feature values.

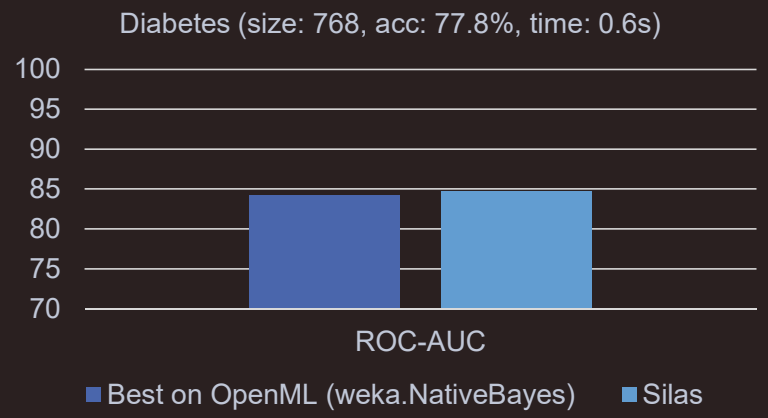
All “parameters” in a formula form a multi-dimensional Euclidean space.

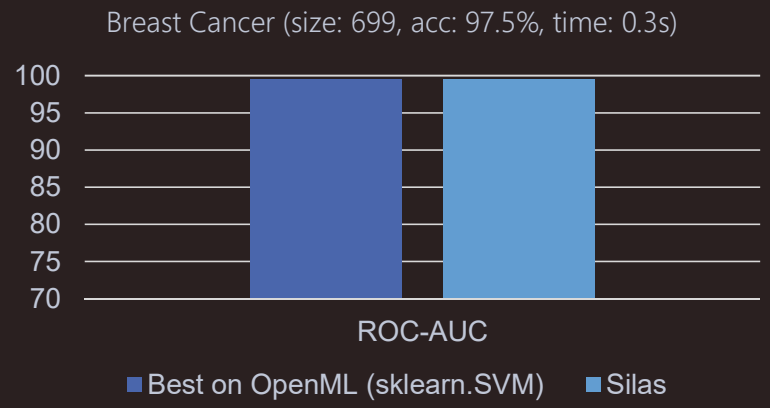
Generation of a formula is a combinatorial optimisation.

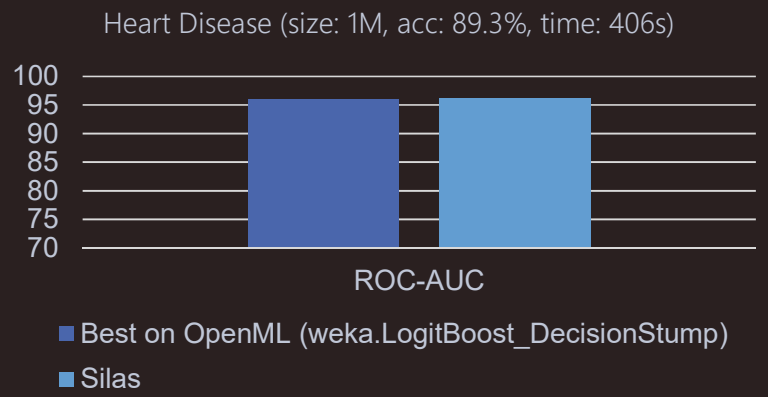
- (Distributed) Monte-Carlo search.
- Particle swarm.
- Genetic algorithm.
- …

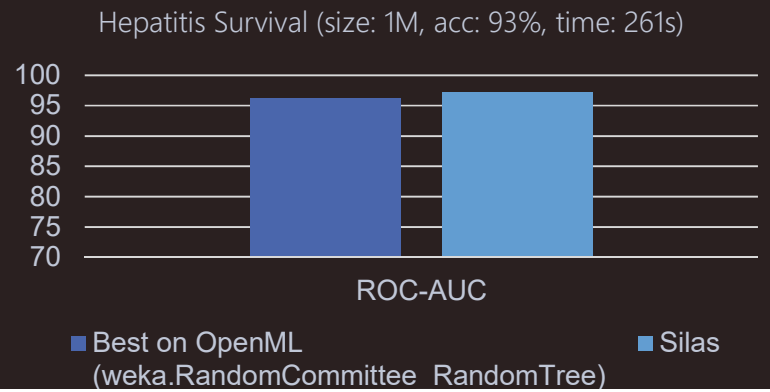
Silas: State-of-the-art classification/prediction results

Silas consistently gives state-of-the-art results in public datasets and benchmarks.
The advantage is especially obvious when dealing with large volumes of data.
Below are examples in the medical domain.









Existing Random Forest Tools

Experiment on building 500 trees for the 1 million flight data set.

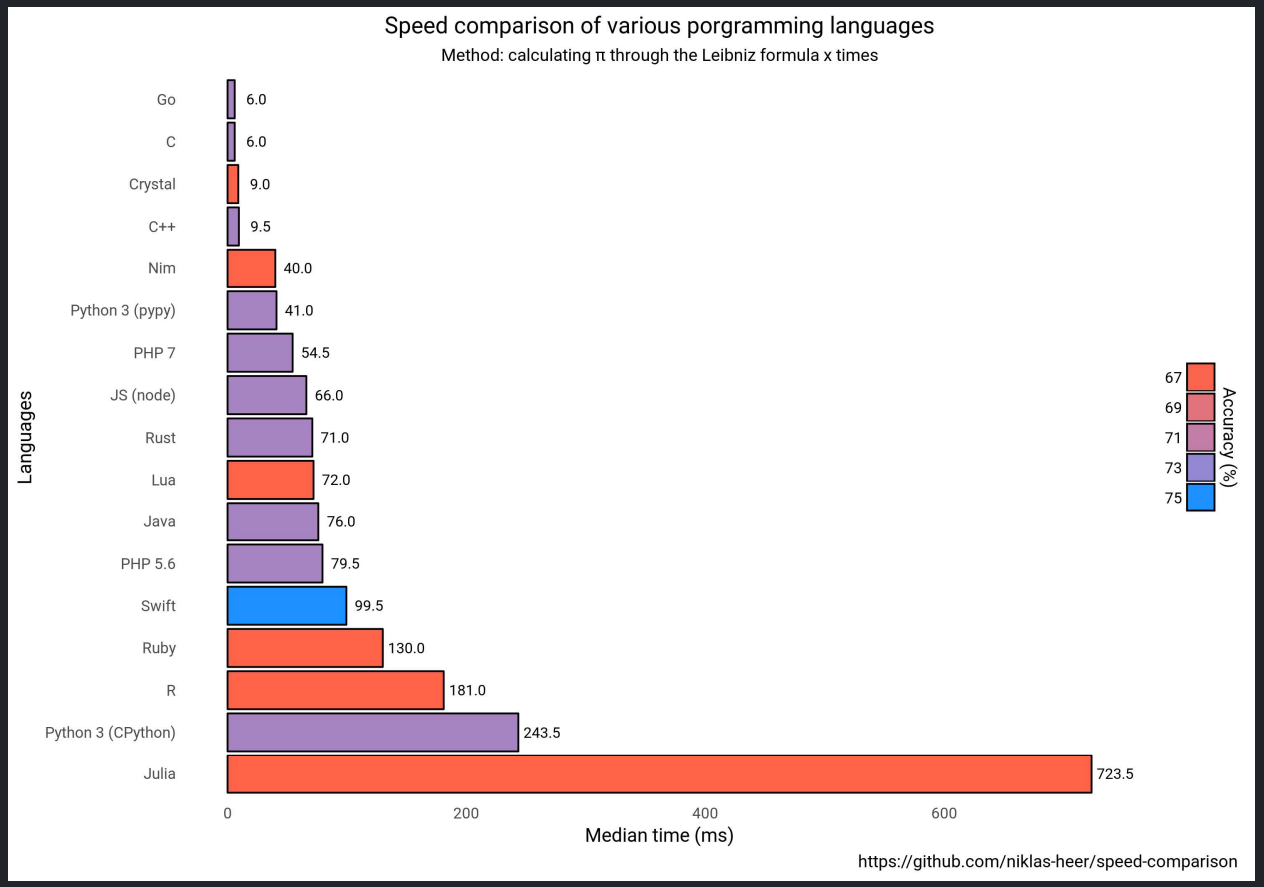
Setup: Amazon EC2 c3.8xlarge instance (32 cores, 60GB RAM).

https://github.com/szilard/benchm-ml

Tool	Time (s)	RAM (GB)	AUC
R	Crash	Crash	N/A
Python	900	20	73.2
H2O	600	5	75.5
Spark	Crash/2000	-	71.4

Machine Learning Tools

Data scientists' favourite tools/languages: Matlab, R, Python, Java,…

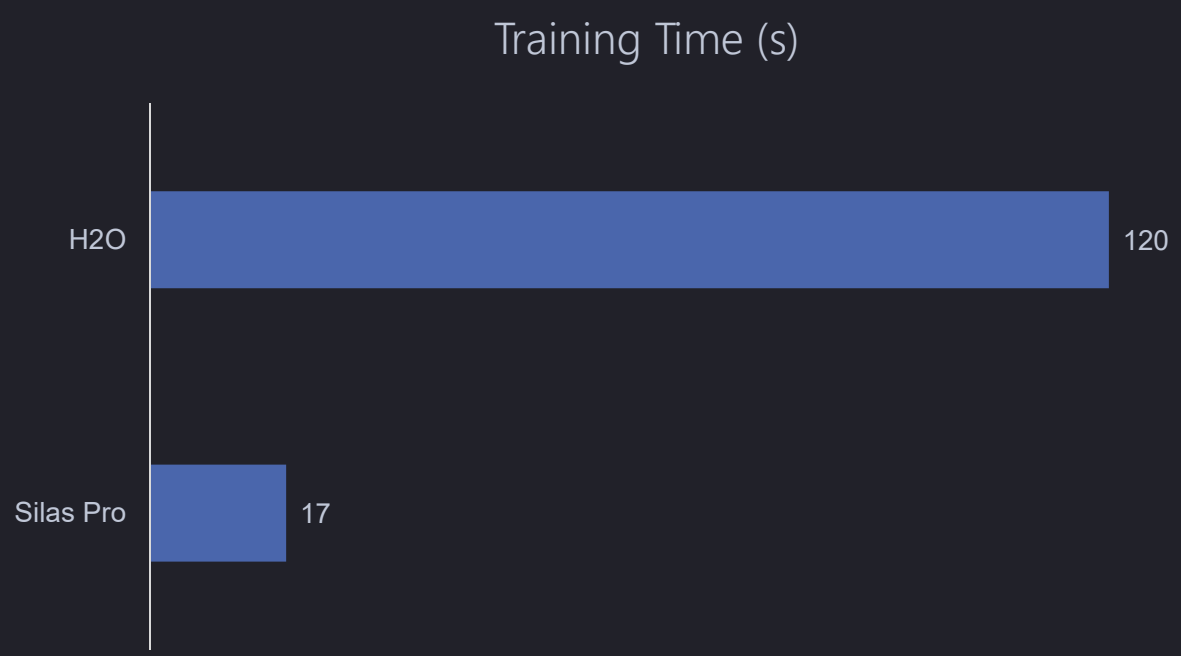


https://github.com/niklas-heer/speed-comparison

High Performance Computing - Speed

To train a model of similar size that yields similar predictive performance,

Silas Pro is **7x faster** than industry leaders.



High Performance Computing – Memory Usage

Silas Pro uses **75% less resources** than competitors.

Data set size/RAM usage	R	Python	H2O	Spark	Silas Pro
1M instances	Crash	20GB	5GB	Crash	**0.8GB**
10M instances	Crash	Crash	25GB	Crash	**5.9GB**

Optimisation Example 1

Entropy vs Gini impurity.

35

Gini impurity and Information Gain Entropy are pretty much the same. And people do use the values interchangeably. Below are the formulae of both:

1. $Gini: Gini(E) = 1 - \sum_{j=1}^{c} p_j^2$
2. $Entropy: H(E) = -\sum_{j=1}^{c} p_j \log p_j$

Given a choice, I would use the Gini impurity, as it doesn't require me to compute logarithmic functions, which are computationally intensive. The closed form of it's solution can also be found.

> Which metric is better to use in different scenarios while using decision trees ?

The Gini impurity, for reasons stated above.

So, **they are pretty much same when it comes to CART analytics.**

Helpful reference for computational comparison of the two methods

Optimisation Example 1

Entropy vs Gini impurity.

FYL2XP1 — Compute y ∗ log2(x +1)

Opcode	Instruction	64-Bit Mode	Compat/Leg Mode	Description
D9 F9	FYL2XP1	Valid	Valid	Replace ST(1) with ST(1) ∗ $\log_2$(ST(0) + 1.0) and pop the register stack.

Description ¶

Computes (ST(1) ∗ $\log_2$(ST(0) + 1.0)), stores the result in register ST(1), and pops the FPU register stack. The source operand in ST(0) must be in the range:

$-(1-\sqrt{2}/2))$to$(1-\sqrt{2}/2)$

The source operand in ST(1) can range from −∞ to +∞. If the ST(0) operand is outside of its acceptable range, the result is undefined and software should not rely on an exception being generated. Under some circumstances exceptions may be generated when ST(0) is out of range, but this behavior is implementation specific and not guaranteed.

Optimisation Example n

Delve into programming languages.

> “No matter what language you work in, programming in a functional style provides benefits. You should do it whenever it is convenient, and you should think hard about the decision when it isn't convenient.”
>
> - John Carmack

Optimisation Example n

http://www.gamasutra.com/view/news/169296/Indepth_Functional_programming_in_C.php

Pure function: concerned with the parameters passed into it and the returned values.

No side effects.

- Doesn't update global state.
- Doesn't maintain internal state.
- Doesn't perform any IO.
- Doesn't mutate input parameters.

Reusability: no environmental assumptions. Easy to port.

Testability: always give the same result for a set of parameters no mater when it's called.

Completely thread safe.

Optimisation Example n

Dynamic (virtual calls) dispatch vs static (curiously recurring template pattern, CRTP) dispatch.

https://eli.thegreenplace.net/2013/12/05/the-cost-of-dynamic-virtual-calls-vs-static-crtp-dispatch-in-c

In some situations, dynamic dispatch is much slower.

- Extra indirection (pointer dereference) for each call to a virtual function.
- Virtual functions usually can't be inlined.
  - Significant cost hit for some small functions which are called often.
- Additional pointer per object.
  - L1 data cache miss more often when there are many objects.

Ensemble tree learning involves

- Hundreds of trees
- Millions (or many more) of nodes in each tree

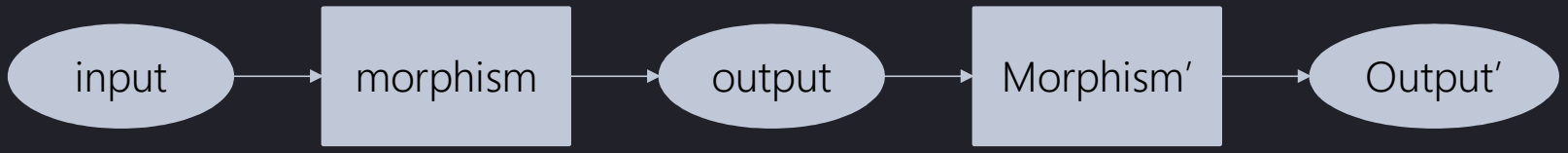
The cost of fetching data in memory is order of magnitudes higher than the cost of computation.

Optimisation Example n

Solution: Template meta programming + functional style programming in C++.

Basic unit of computation: morphism (pure function with CRTP). 0 run-time overhead

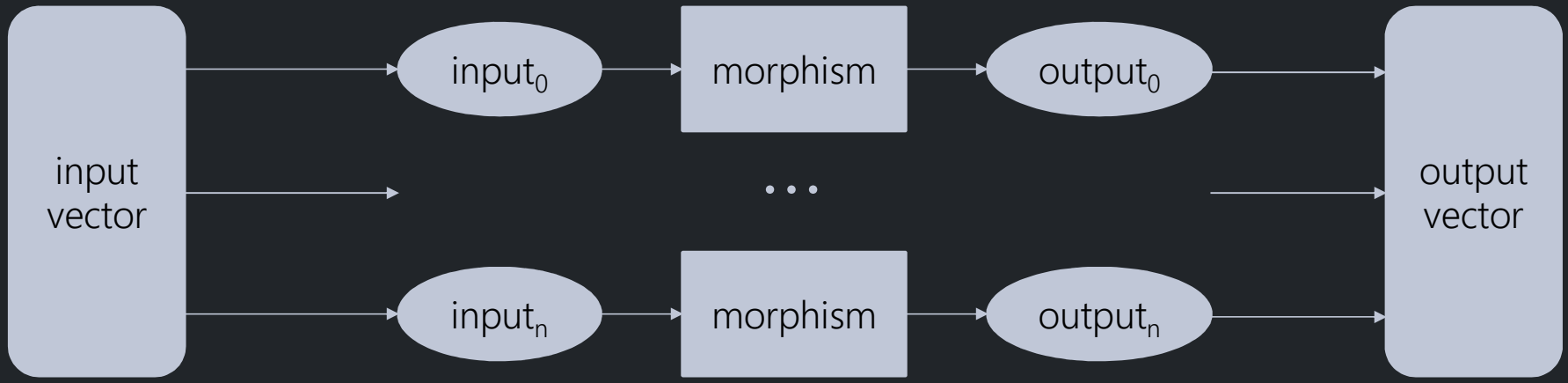
Sequential composition.



Optimisation Example n

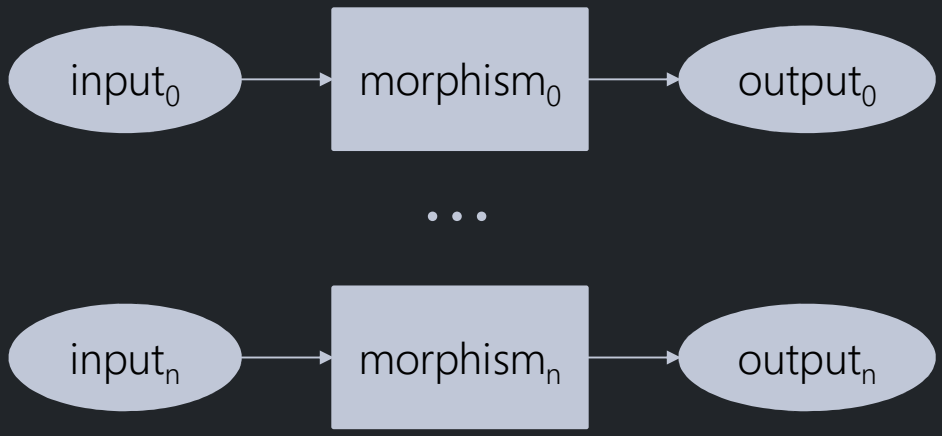
Repeat.

Batch processing.



Optimisation Example n

Parallel computation.



Lambda function.

Data Storage – Stable Vector

Normal vectors (`std::vector`) in C++ are often subject to memory reallocations.

- Wastes a ton of time in some situations.
- For decision trees: each leaf node is associated with a vector of indices to data points.
- Iterating over data in non-contiguous memory incurs cache-miss.
- Compiler and CPU are unable to perform effective cache prefetching.

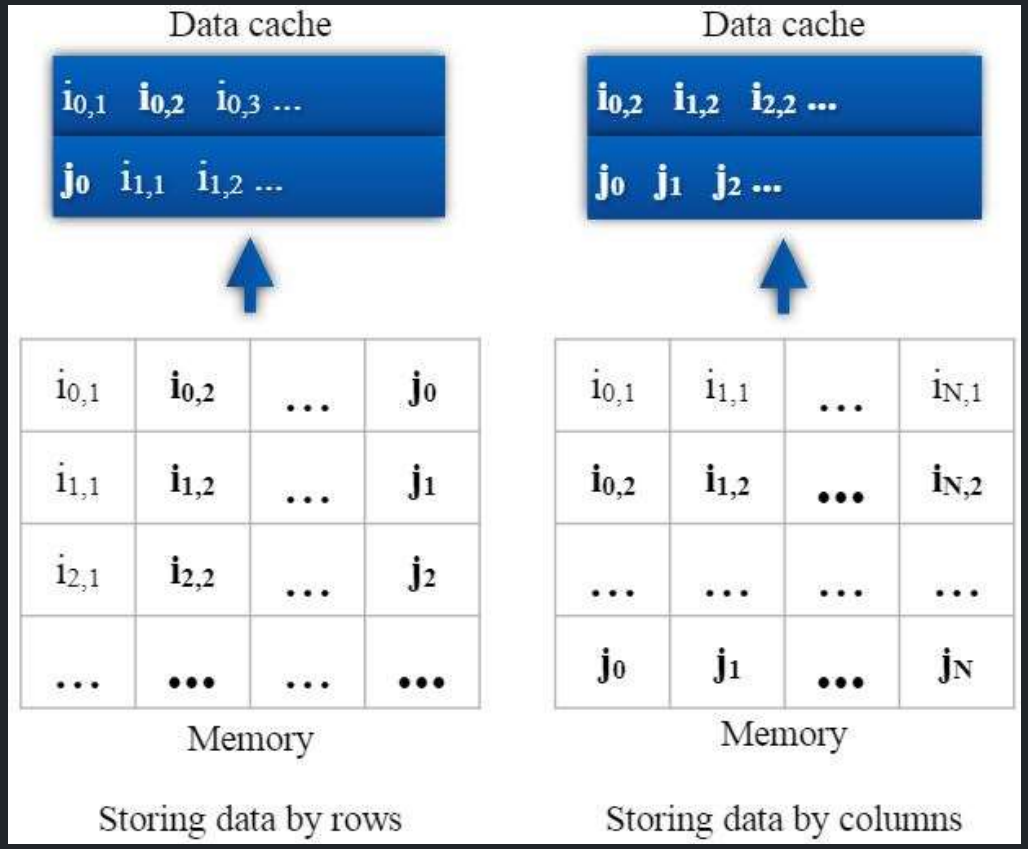
We have implemented our own stable vector for stoing data.

- Mechanisms for allocating memory pages.
- Storing multi-dimensional data into a flat array in contiguous memory segments.
- Access method via custom memory address arithmetic.
- Stable referencing.

Only store references to the original data points in leaf nodes using stable vector.

- Significant speed improvement!

Data Storage – Storing Data By Columns



High Information Density

- Automatically detect and use the smallest data type for each feature.
- Outcome for binary classification is defined as a Bool variable.
- Nominal feature with less than 256 distinct values is defined as a `uint_8` variable.

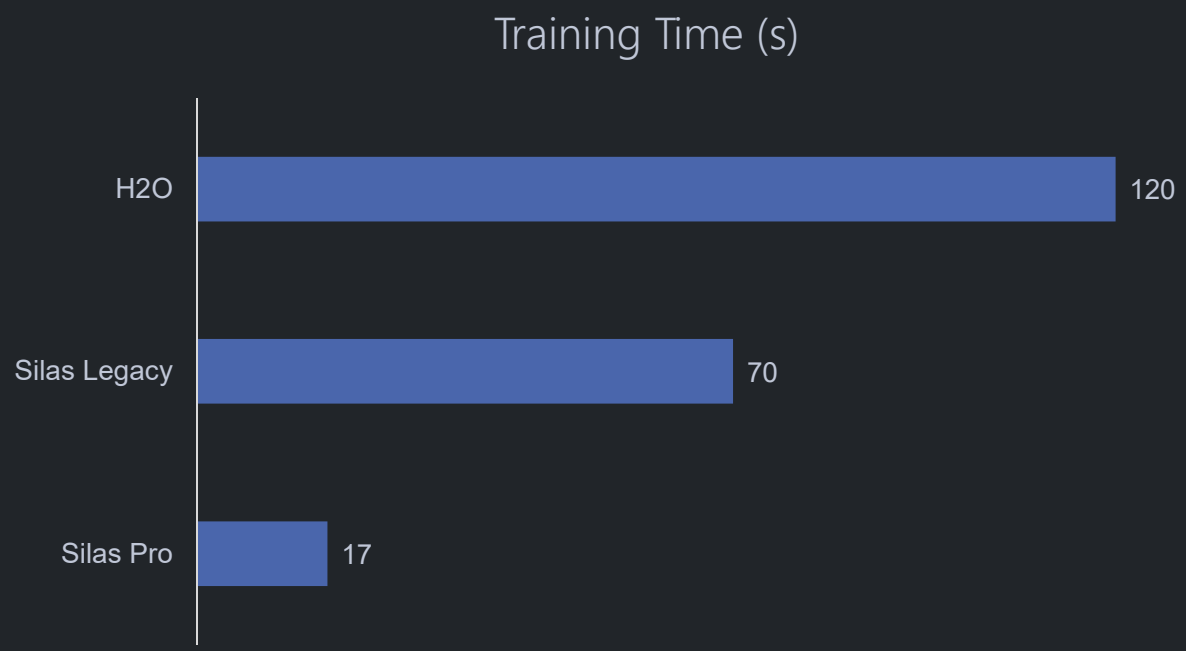
1 cache-miss per data point.

1 cache-miss at first, then 1 cache-miss per (size of cache line) / (size of value).

High Performance Computing

To train a model of similar size that yields similar predictive performance,

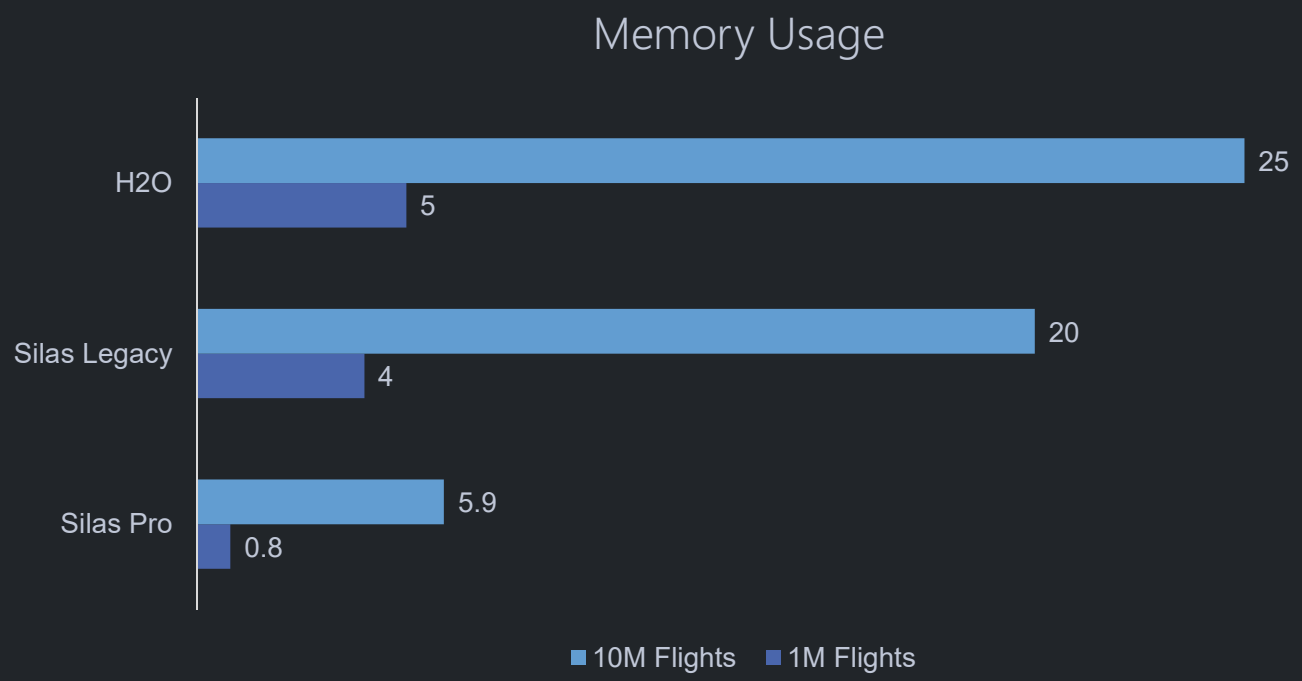
Silas Legacy Code is **1.7x faster** than industry leaders.
Silas Pro is **7x faster** than industry leaders.



High Performance Computing

To train a model of similar size that yields similar predictive performance,

Silas Legacy Code uses **20% less** memory than industry leaders.
Silas Pro is **75% less** memory than industry leaders.



Big Data And Security

"There is therefore a need to better understand all the implications and intricacies of AI and machine learning applications in finance – as well as data privacy, conduct risks and cybersecurity."

- Andrew Hobbs

AI Failures

- Complex AI stock trading software caused a trillion dollar flash crash.
- Microsoft chabot Tay became racist and bigoted.
- Amazon's Alexa offered porn to child.
- Self-driving car had a deadly accident.
- AI designed to predict crime acted racist.
- ...

People don't understand why they happened.

Trust Issues in AI/ML

"Many of the models that result from the use of AI or machine learning techniques are difficult or impossible to interpret. The lack of interpretability may be overlooked in various situations, including, for example, if the model's performance exceeds that of more interpretable models."

- Financial Stability Board

"Financial compliance standards don't make AI research impossible for companies like Morgan Stanley, but they do make innovation either slower or more expensive than in lighter-regulated industries."

- Pegah Ebrahimi, COO of Global Technology Banking at Morgan Stanley

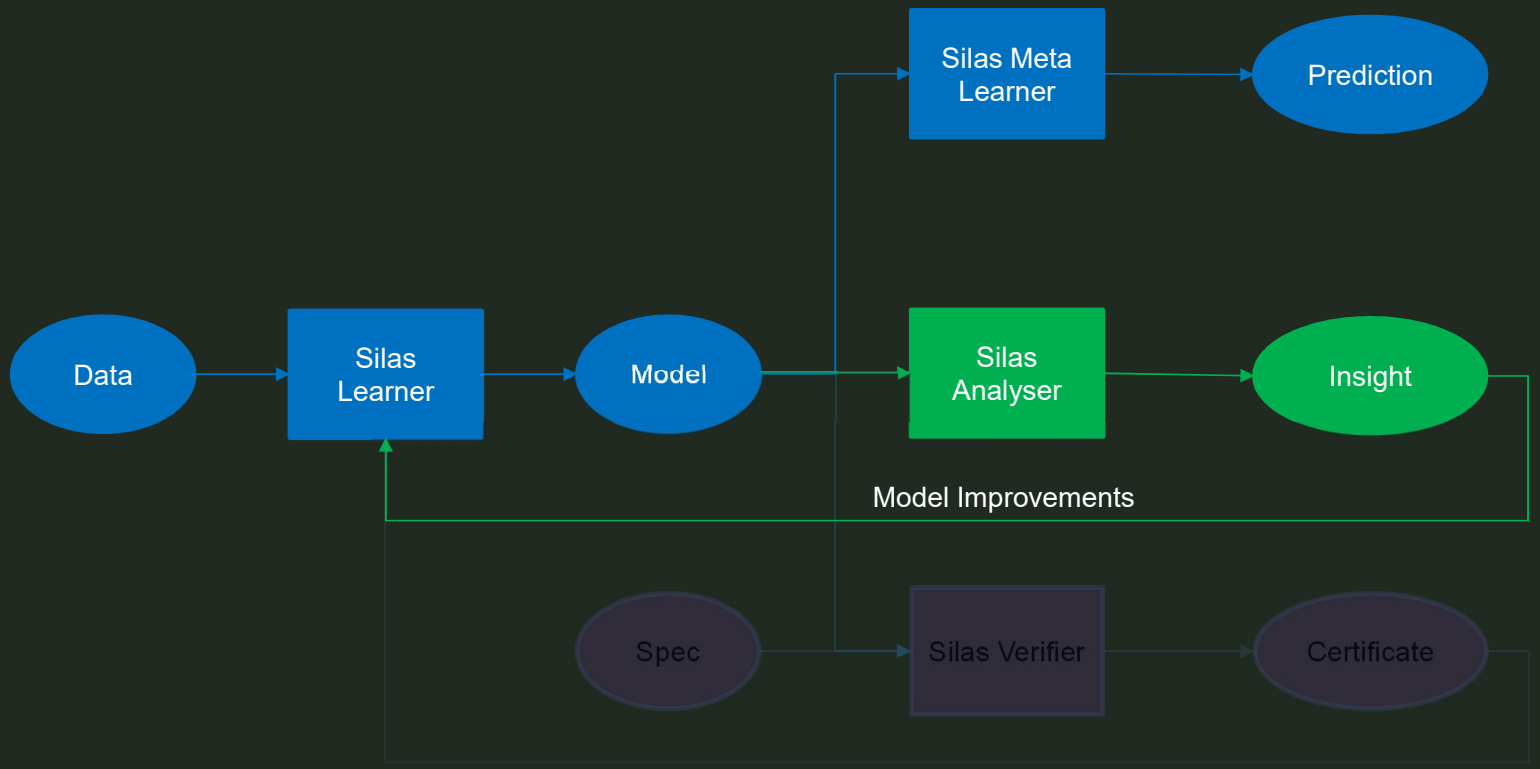
Whitebox Machine Learning?

Explainability
- Understand how the prediction model works.
- Understand how the prediction on an instance works.

Verifiability.
- Ensure that the model does not do bad things.
- Ensure that the model does good things.

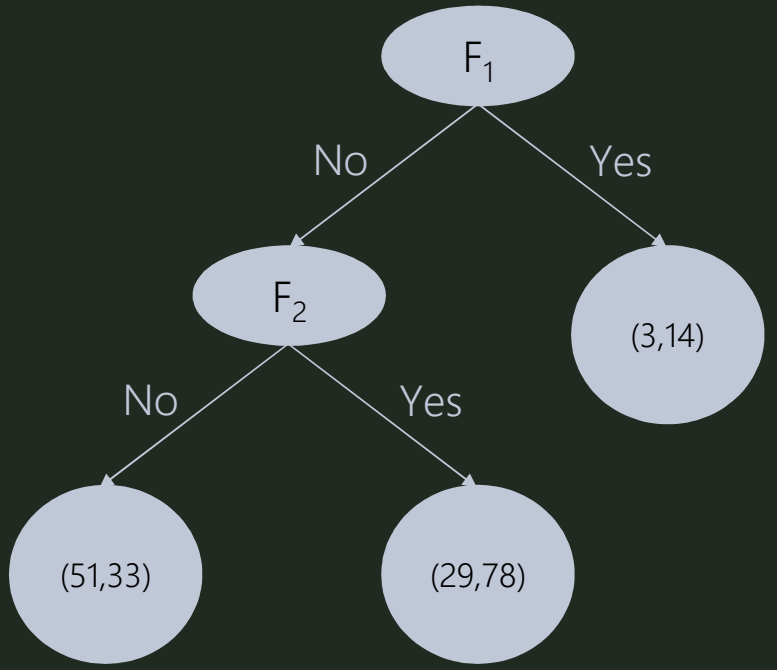
Interactability.
- Improve/correct the model.

Silas Model Insight



Formula Extraction

- Node formula: Logical formulae with arithmetic, comparison, membership etc.
- Branch Formula: Conjunction of every (negated) node on a branch implies a class.
- Tree Formula: Disjunction of the branch formulae.
- Each formula has a weight.
- Example:
- Example:
- Node formula: F_1, F_2
- Branch formulae:
  - $\neg F_1 \wedge \neg F_2 \Rightarrow$ Class 0
  - $\neg F_1 \wedge F_2 \Rightarrow$ Class 1
  - $F_1 \Rightarrow$ Class 1
- Exploited sampling for analysis:
  - Sample a subset of nodes in a branch.
  - Sample a subset of branches in a tree.
  - Sample a subset of trees in a forest.



Model Insight

- How does the model work in general?
- Minimal Unsatisfiable Core (MUC)
  - Inconsistencies among the decision trees.
  - Use the MUC to improve the trees.
  - Use the MUC as weak classifier in boosting.
- Maximum Satisfiable Subset (MSS)
  - The common ground of the decision trees.
  - Use the MSS to find key features.
  - Use the MSS to explain the rationale of the decision making.

Prediction Insight

- How does the model derive prediction for a given new data point?
- Similar to Model Insight.
- Symbolic execution of the new data point on each tree.
- Obtain an execution path (a branch) on each tree.
- Customised solver.

Trade-offs In Sampling

- Sampling too much (or take all the formulae):
  - Even the best SMT solver with parallel analysis algorithm takes forever.
  - The reasoning result gives very narrow conditions.
  - Example: $32 < \text{Age} \leq 33$ leads to positive diabetes.
- Sampling too little:
  - The reasoning result may not represent the most important decision-making.
  - The result gives very general conditions.
  - Example: $15 < \text{Age} \leq 68$ leads to positive diabetes.

Observation

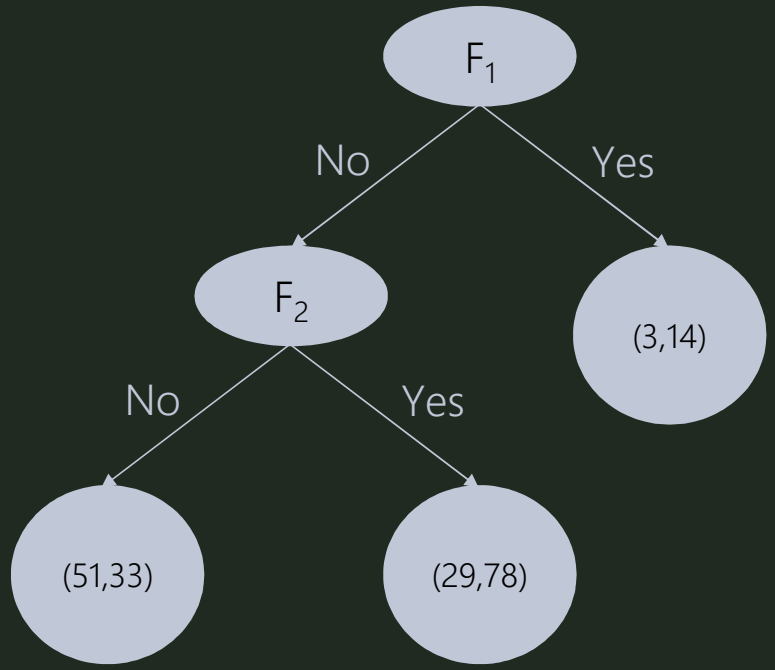
Insight is concerned with the conditions that lead to the prediction.

The fact that

$$\neg F_1 \wedge \neg F_2 \Rightarrow \text{Class 0 and } \neg F_1 \wedge F_2 \Rightarrow \text{Class 1}$$

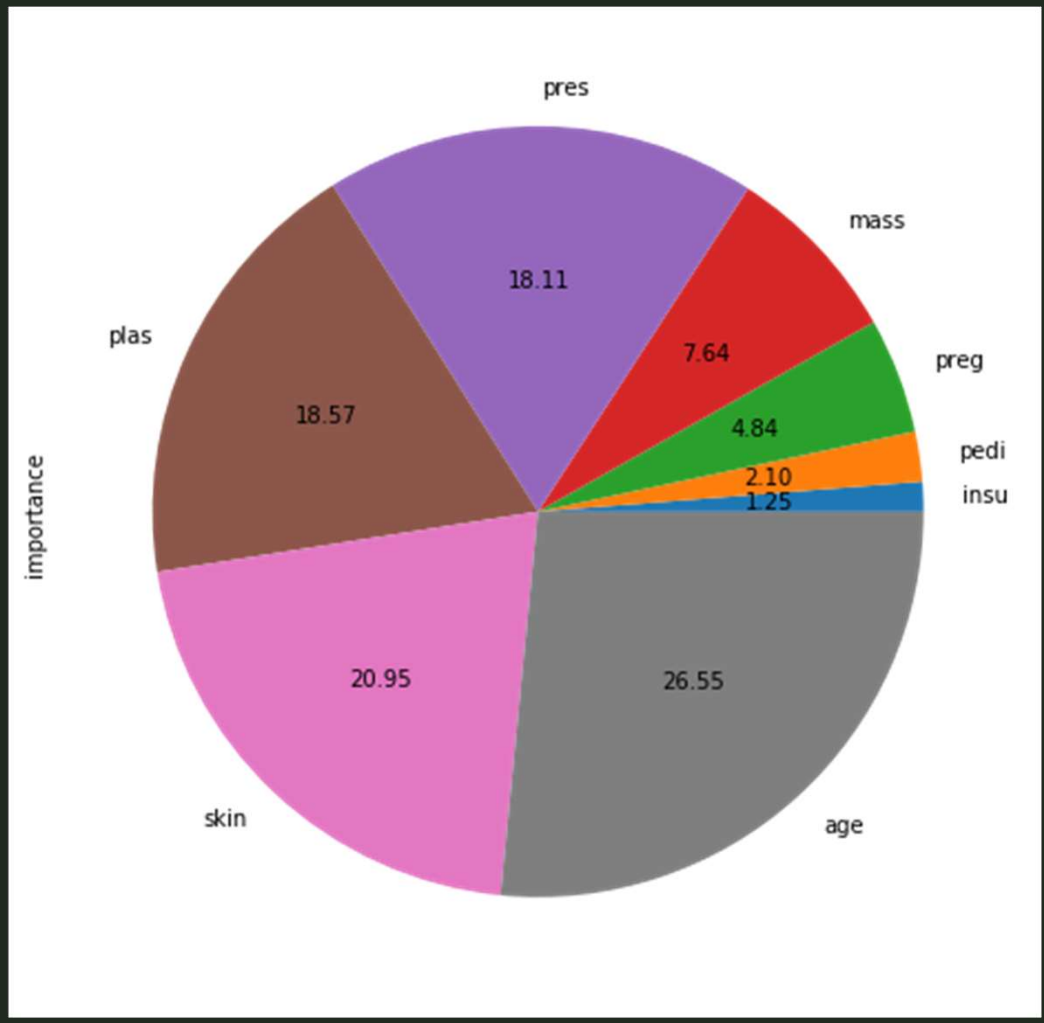
are extracted from the same tree is not critical…

if we can store the weight of the tree.



Case Study: Diabetes

Model Insight – Feature Importance

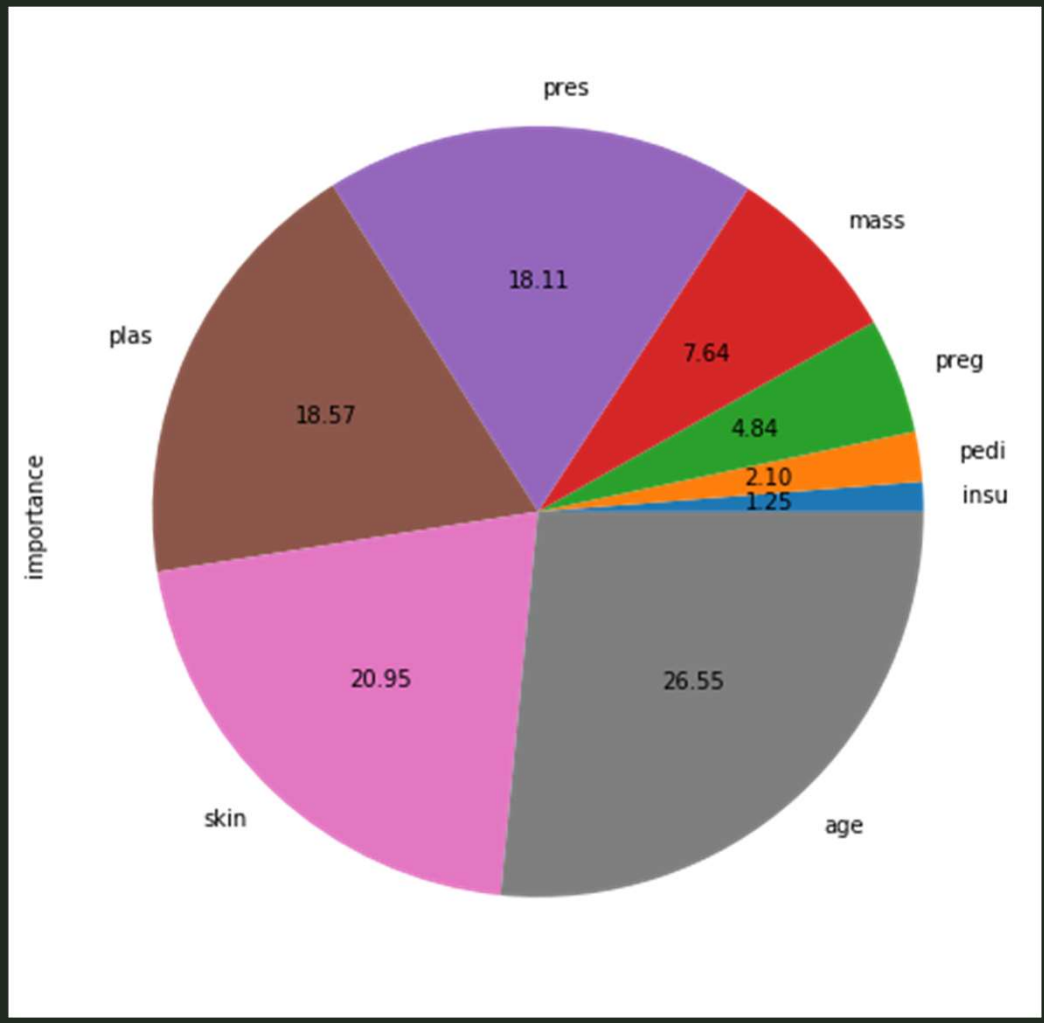


Case Study: Diabetes

Model Insight – Decision Making

Positive Diabetes:

- 30 ≤ age < 47
- 31 ≤ skin < 99
- 155 ≤ plas < 157
- 40 ≤ pres < 122
- 30 ≤ mass < 40.8

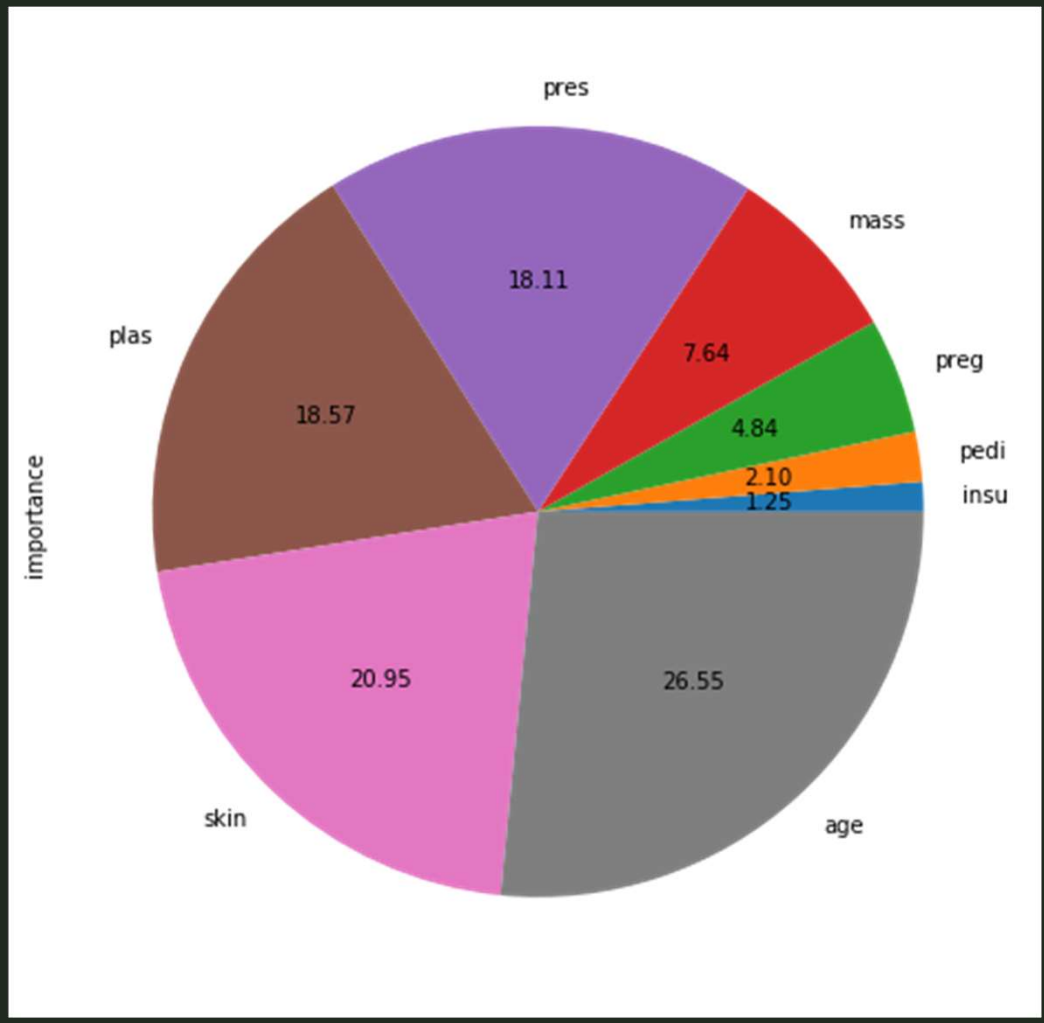


Case Study: Diabetes

Model Insight – Decision Making

Negative Diabetes:

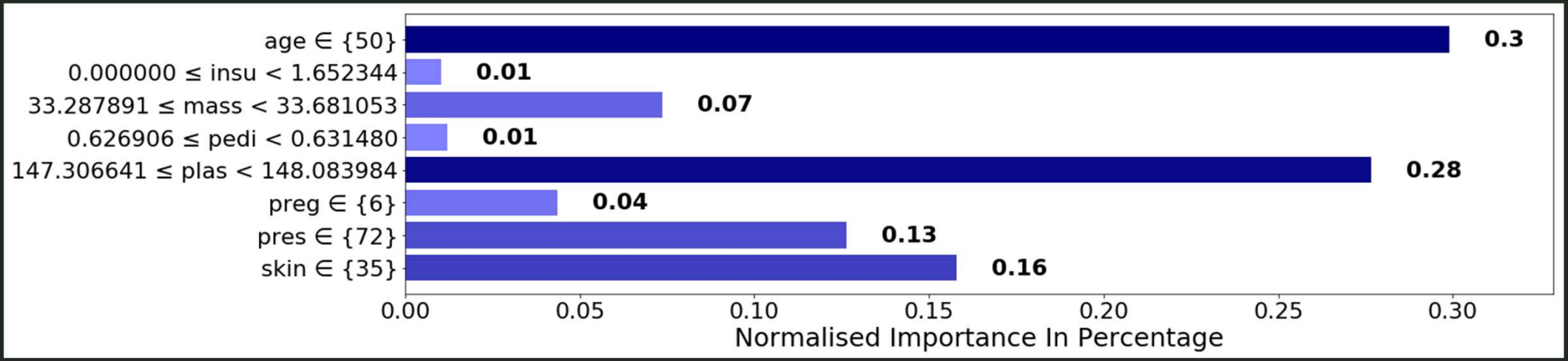
- 24 ≤ age < 25
- 0 ≤ skin < 30
- 114 ≤ plas < 117
- 0 ≤ pres < 68
- 26.4 ≤ mass < 26.8



Case Study: Diabetes

Prediction Insight (positive case)

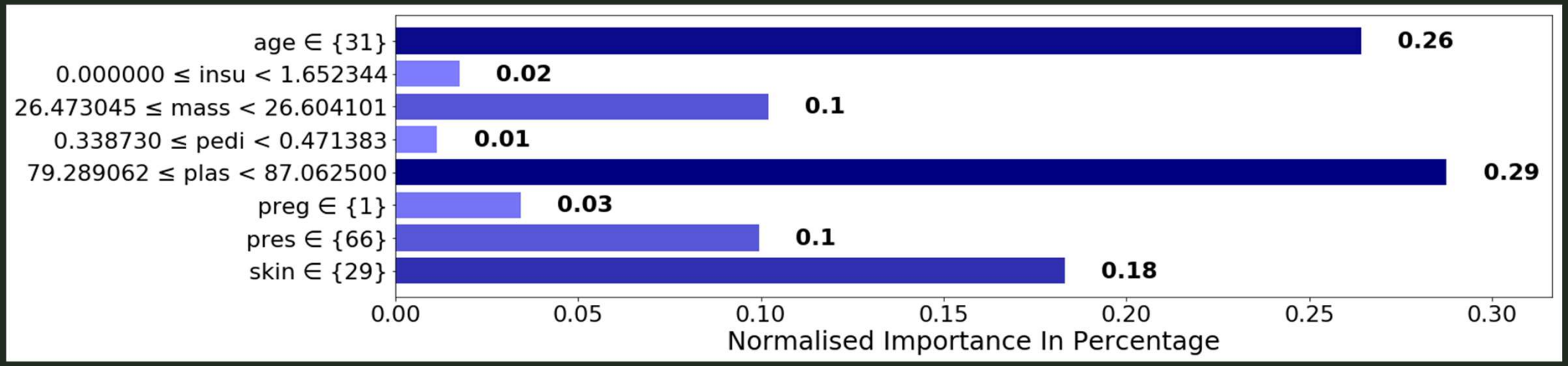
Predicted probability: 0.8581



Case Study: Diabetes

Prediction Insight (negative case)

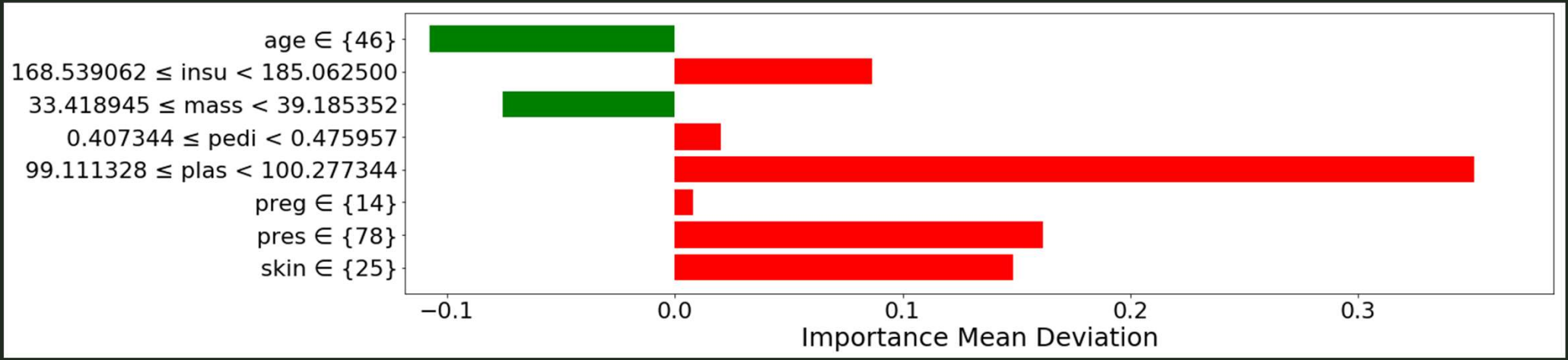
Predicted probability: 0.3383



Case Study: Diabetes

Prediction Insight (positive case)

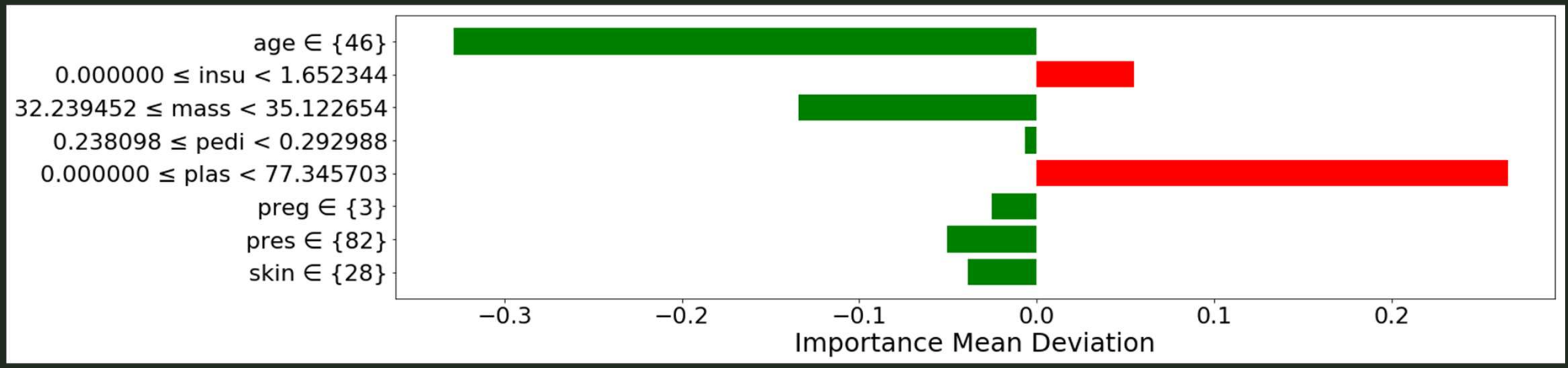
Predicted probability: 0.6296



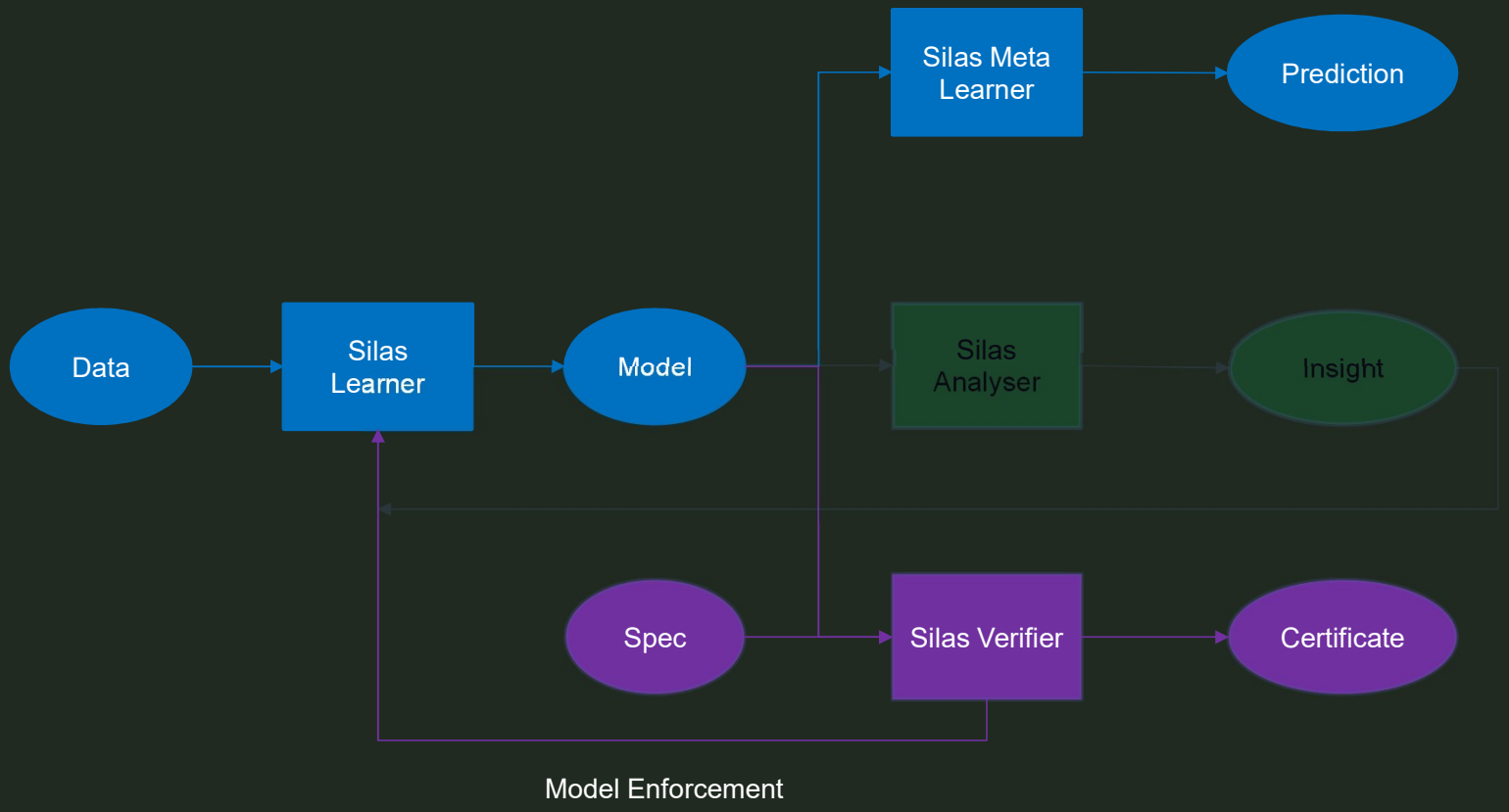
Case Study: Diabetes

Prediction Insight (negative case)

Predicted probability: 0.3637



Silas Model Audit



Model Audit

- User-defined constraints
  - E.g., BMI > 45 implies positive diabetes.
- Use SMT solving to verify the model against constraints.
  - F is a tree formula.
  - C is a user-defined constraint.
  - Check satisfiability of $\neg (F \Rightarrow C)$. If unsat, then C is valid in F.
- Model Enforcement.
  - Build decision trees that are guaranteed to satisfy user-defined constraints.
  - And have good predictive performance.

Strong Verification

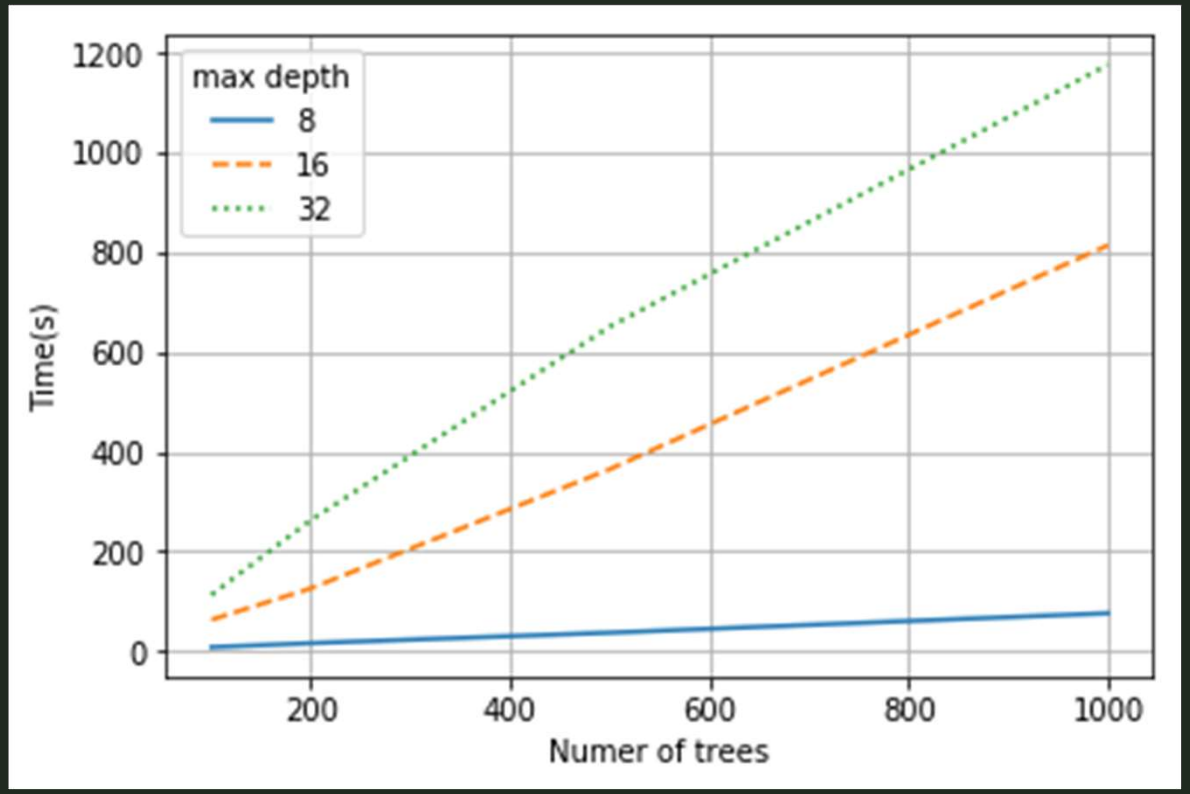
Verify each tree independently, output the list of trees that do not satisfy user requirements.

Theorem (Soundness)

If the verification outputs positive, then the ensemble tree model is correct w.r.t. user requirements.

It is possible that the ensemble tree model is correct but certain individual trees do not meet user requirements.

Scalability



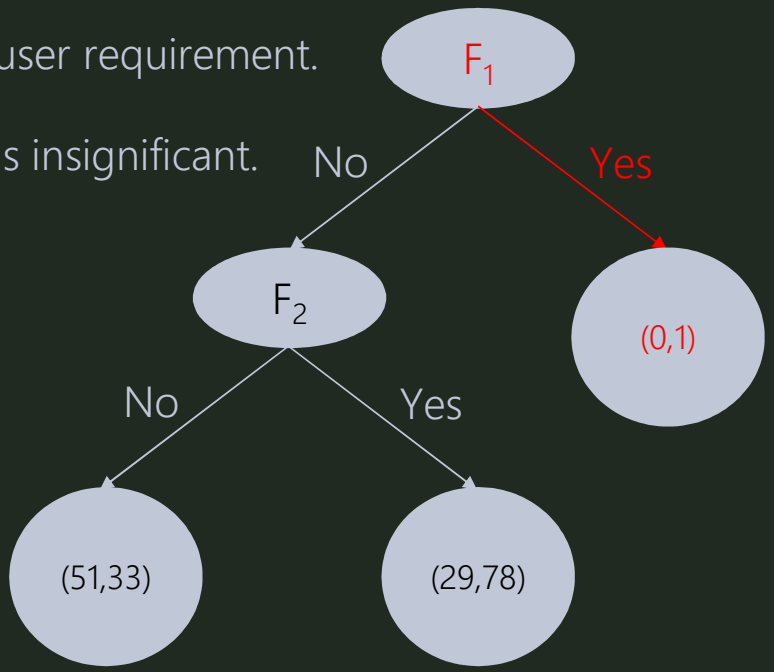
Strong Verification

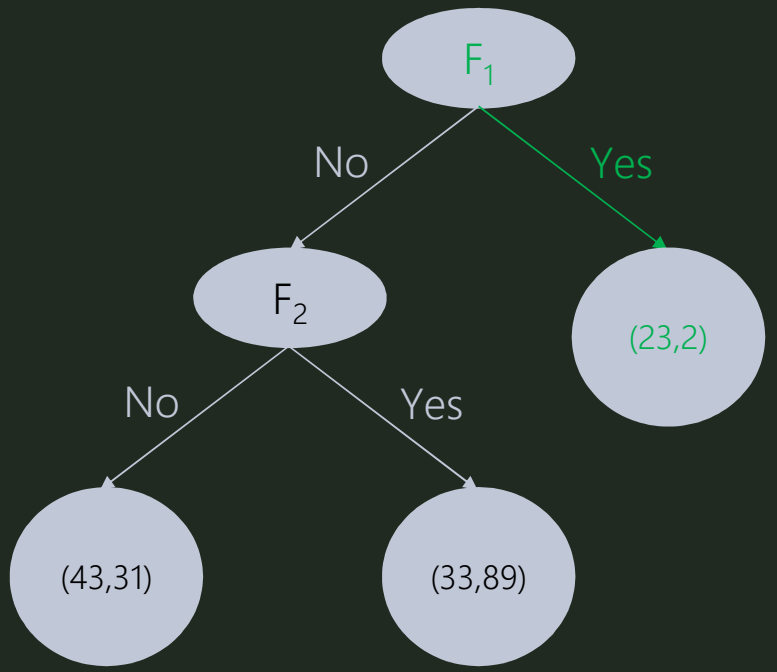
It is possible that the ensemble tree model is correct but certain individual trees do not meet user requirements.

User requirement: Class = 1 ⇒ ¬ F_1

The left tree violates the user requirement.

But the votes at the leaf is insignificant.





The ensemble tree model will never violate the user requirement.

Problem: How To Merge Decision Trees?

Solution 0: merge all the branches from every tree.

The resultant tree will be exponentially larger than each individual tree.

Hope: there will be a lot of combined branches which are equivalent to false.

- Data points will never go through these branches, so we can remove them.

The transform will still take a looooooong time.

There will not be enough memory to hold the data.

Write to hard drive more often?

- Even slower.

Distributed computing?

- Expensive.

Problem: How To Merge Decision Trees?

Solution 1: change the voting system.

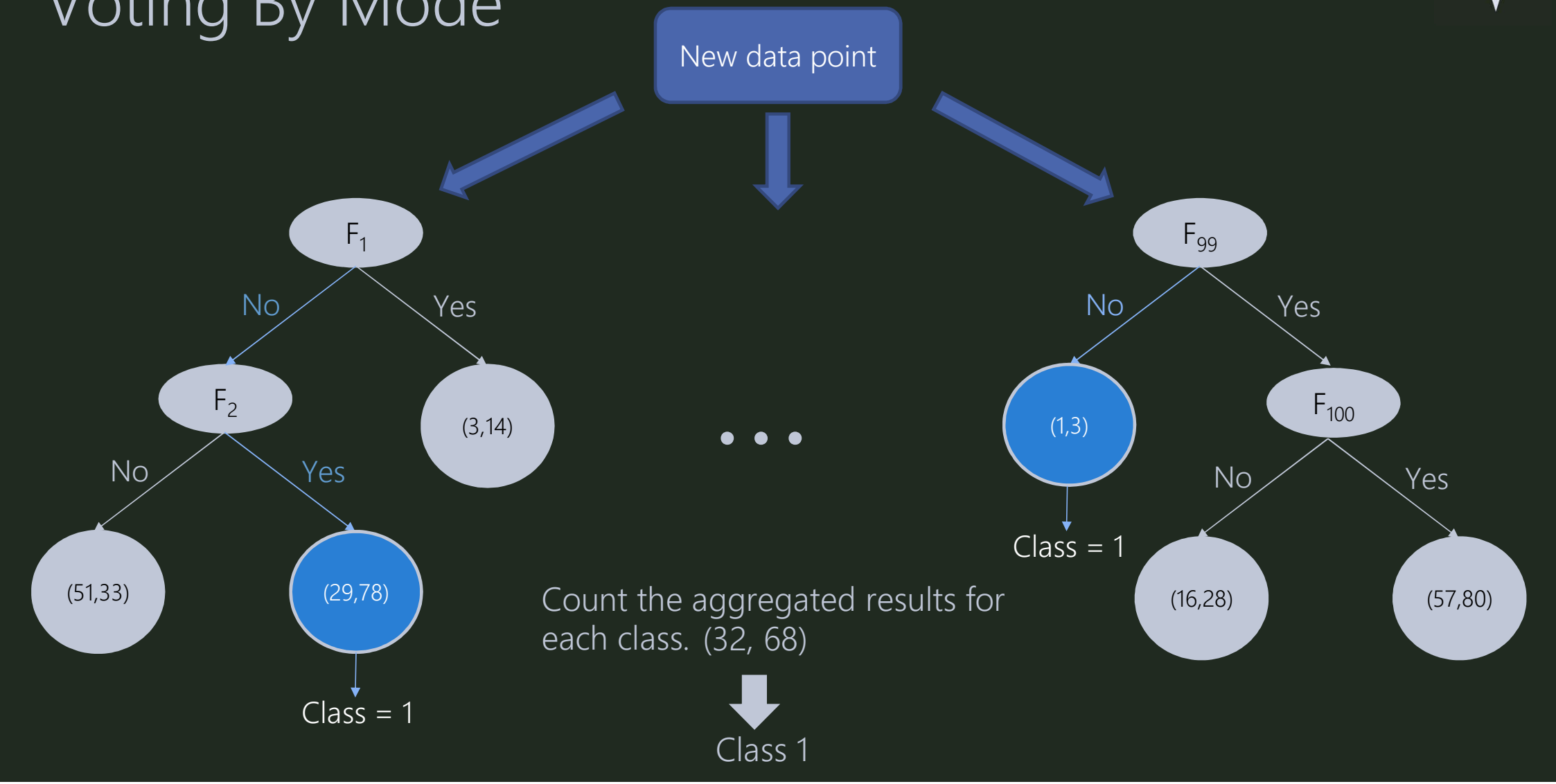
Voting by distribution (VD)

- Each tree outputs the distribution of different classes.
- Aggregate the distribution over all the trees.

Voting by mode (VM)

- Each tree outputs the voted class.
- Aggregate the count for voted classes over all the trees.

Voting By Mode



Soundness Results

Voting Method	Binary Classification	Multi-class Classification
VM	Sound	Sound
VD	Sound	Not Sound

N.B. Voting by mode yields slightly worse predictive performance in our experiment.

Enforcement Learning

What if the prediction model does not satisfy certain standards or regulations?

Train a model that is correct-by-construction.

- Guaranteed to satisfy all user defined constraints.
- Good predictive performance.