

Design of a diagnosis and follow-up platform for patients with chronic headaches

Kiani Lannoye & Gilles Vandewiele

Faculty of Engineering and Architecture

Intro

Machine learning - DT's

Platform requirements

Genetic merging of DT's

Mobile application

Visualization

Backend and data exposure

Conclusion & future work

Intro

Current process UH Ghent

Platform requirements

Mobile application

Backend and data exposure

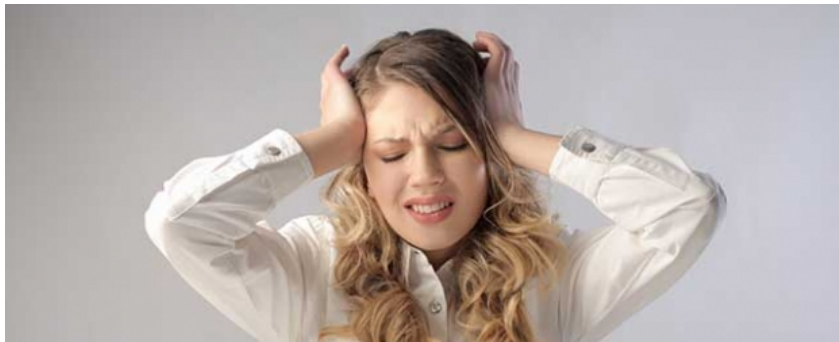
Machine learning - DT's

Genetic merging of DT's

Visualization

Conclusion & future work

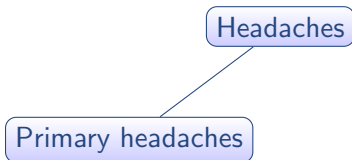
Introduction



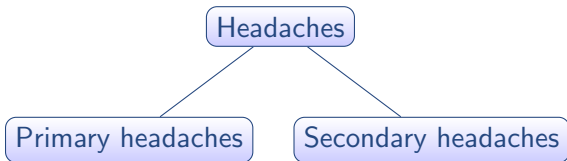
Headaches

Headaches

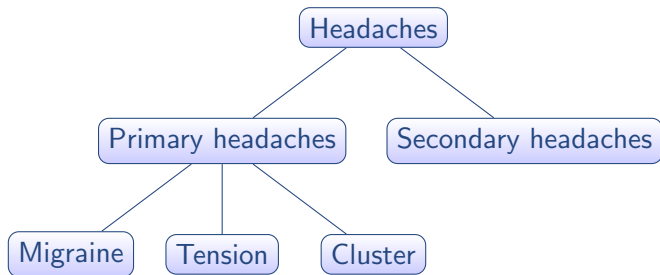
Headaches



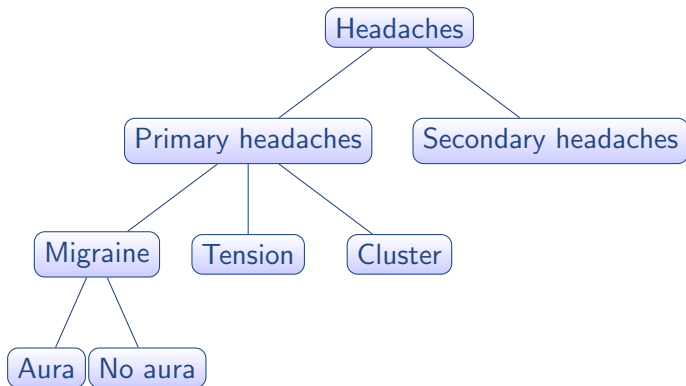
Headaches



Headaches



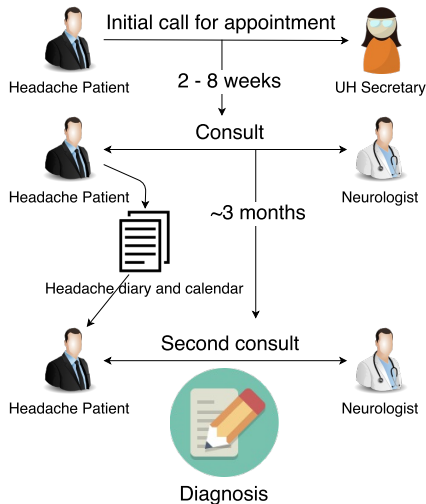
Headaches



Current process UH Ghent

Current process at UH Ghent is:

- ▶ Not digital
- ▶ cumbersome
- ▶ long-lasting



So there is need for a better (digital) alternative! This alternative has to:

- ▶ capture at least the same information as current solution
- ▶ be more efficient
- ▶ provide a second opinion for the doctors (auto-diagnose)

Intro

Machine learning - DT's

Platform requirements

Genetic merging of DT's

Mobile application

Visualization

Backend and data exposure

Conclusion & future work

Platform requirements

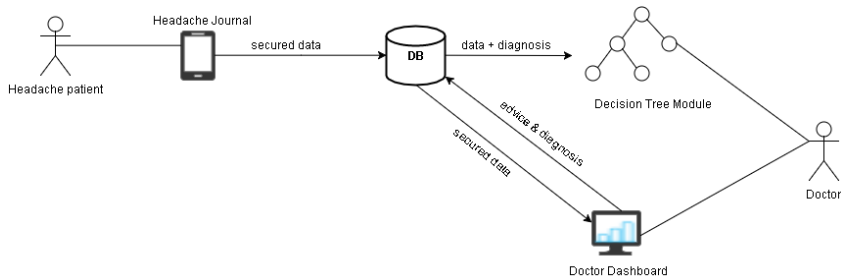
Our proposed alternative consists of:

- ▶ Headache journal: mobile app
- ▶ Doctor Dashboard: web application
- ▶ Machine learning module: auto-classify

Solution non-functional requirements:

- ▶ Security
- ▶ Availability
- ▶ Performance
- ▶ Usability

Platform requirements



Intro

Machine learning - DT's

Platform requirements

Genetic merging of DT's

Mobile application

Development paradigms

Chronicals

Visualization

Conclusion & future work

Backend and data exposure

Mobile Application

Why create a new application?

Competition

- ▶ Migraine Buddy
- ▶ Headache Diary
- ▶ Pfizer headache journal

Mobile Application

Why create a new application?

Competition

- ▶ Migraine Buddy
- ▶ Headache Diary
- ▶ Pfizer headache journal

All good, but:

Mobile Application

Why create a new application?

Competition

- ▶ Migraine Buddy
- ▶ Headache Diary
- ▶ Pfizer headache journal

All good, but:

- ▶ none offers usable data export
- ▶ none captures all data needed

Development paradigms

Different kinds of approaches for mobile application development:

- ▶ Web application
- ▶ Hybrid application
- ▶ Native application

→ How do we choose?

Web application

Webapps are developed once and can be viewed on (almost) all smartphones (via built-in web engine).

- + “write once, run everywhere” \Rightarrow lower cost
- + No installation required
- limited use of device specific features (GPS, camera, ...)
- **Not all devices same web engines \Rightarrow other view**
- No native look and feel

\Rightarrow **No web application**

Native application

Native apps are developed once for each OS and installed on the device.

- + Best performance (optimized machine code at compile time)
- + Device specific features usable (GPS, camera, ...)
- + Native look and feel
- **Write code for each OS (very costly dev + maintenance)**
- Installation required

⇒ Native application?

Hybrid application

Hybrid apps are developed once and installed on the device. It uses the devices internal web engine, but has more control than web applications.

- + “write once, run everywhere” \Rightarrow lower cost
- + Better performance (semi-optimized machine code)
- + Device specific features usable (GPS, camera, ...)
- + Native look and feel (using libraries)
- Installation required
- **Not all devices same web engines \Rightarrow other view (but manageable)**

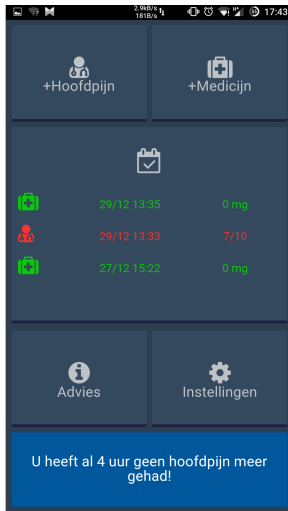
\Rightarrow Hybrid application?

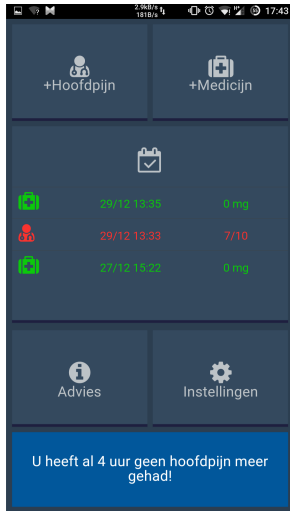
Hybrid vs Native

	Native	Cross-platform
+	<ul style="list-style-type: none"> + Native UX + device-specific features + Better performance 	<ul style="list-style-type: none"> + 1 language + Write once, run everywhere + Less maintenance
-	<ul style="list-style-type: none"> - Multiple languages - Time consuming (development) 	<ul style="list-style-type: none"> - Slower (lower performance) - Less device specific features - Harder to release online (Play Store/App Store)

Hybrid vs Native

	Native	Cross-platform
+	<ul style="list-style-type: none"> + Native UX + device-specific features + Better performance 	<ul style="list-style-type: none"> + 1 language + Write once, run everywhere + Less maintenance
-	<ul style="list-style-type: none"> - Multiple languages - Time consuming (development) 	<ul style="list-style-type: none"> - Slower (lower performance) - Less device specific features - Harder to release online (Play Store/App Store)





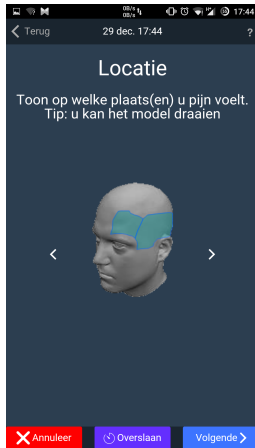
Chronicals



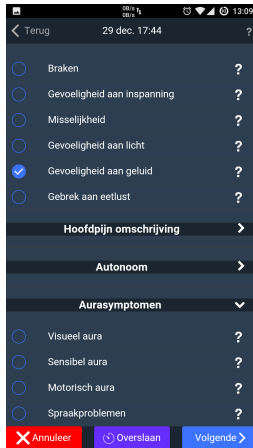
Chronicals



Chronicals

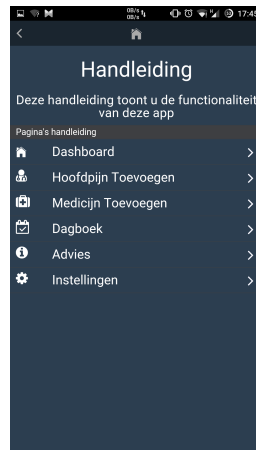
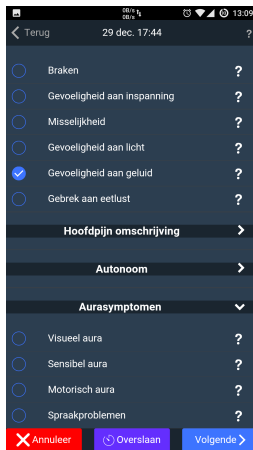
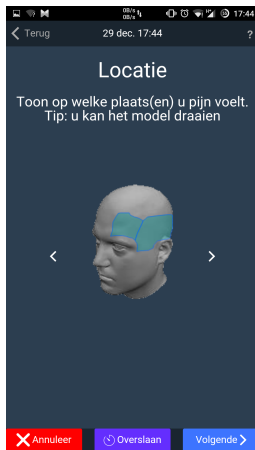


Mobile application



Chronicals

Chronicals



Intro

Machine learning - DT's

Platform requirements

Genetic merging of DT's

Mobile application

Visualization

Backend and data exposure

Conclusion & future work

Intro

Platform requirements

Mobile application

Backend and data exposure

Machine learning - DT's

Genetic merging of DT's

Visualization

Conclusion & future work

Intro

Machine learning - DT's

Platform requirements

Genetic merging of DT's

Mobile application

Visualization

Backend and data exposure

Conclusion & future work

Many different induction algorithms



C4.5 (C5.0)



CART



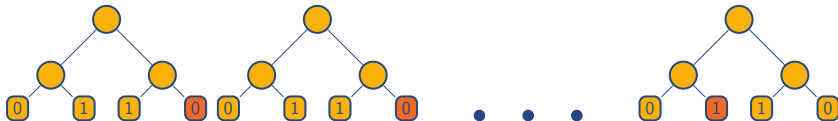
QUEST

...

→ **Which tree is the most beautiful?**

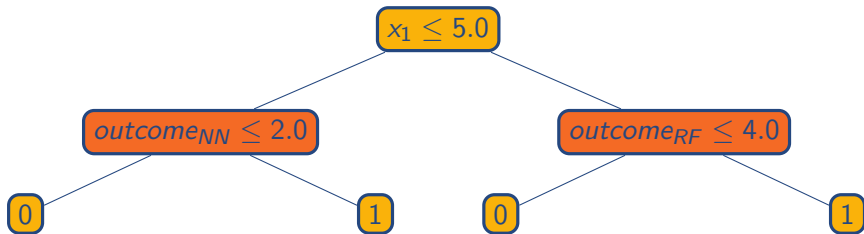
Current ensembles lack interpretability

Boosting, bagging, random forests, etc. require majority voting (classification) or mean calculation (regression) to obtain prediction



Current ensembles lack interpretability

The final decision tree obtained by **stacking** contains uninterpretable internal nodes

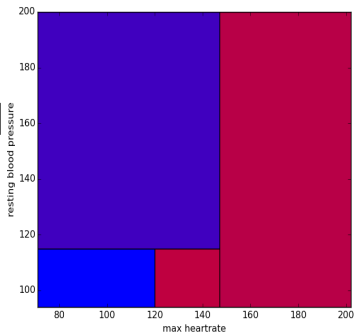
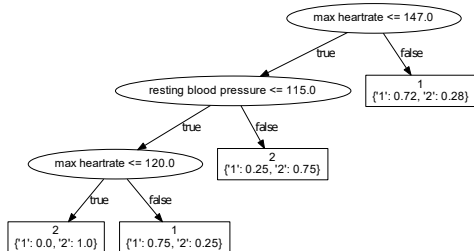


Decision tree \rightarrow decision space

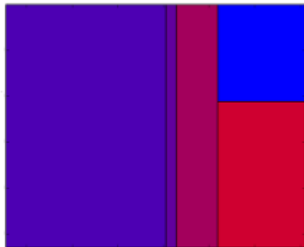
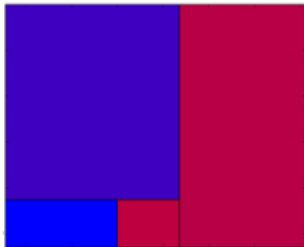
Converting decision trees to decision spaces

We can define a one-to-one mapping between a decision tree and a set of k -dimensional hyperplanes ($k = \text{\#features}$), called **decision space**. Each node in the decision tree corresponds to a hyperplane in the decision space.

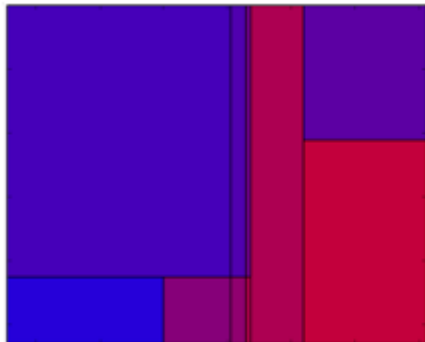
Decision tree → decision space



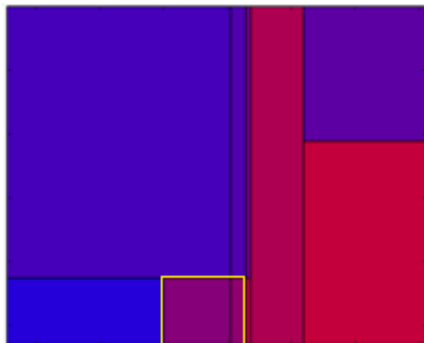
Merging decision spaces



Merging decision spaces



Pruning decision spaces



Decision space \rightarrow decision tree

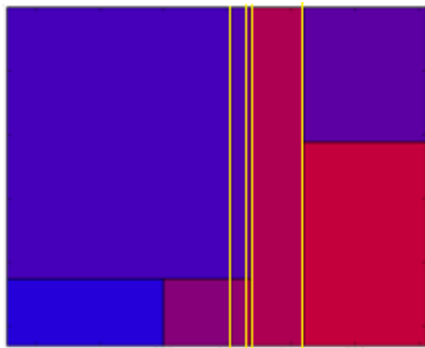
Converting decision spaces to decision trees

One-to-one mapping from decision tree to space is lost during conversion because the order is lost. Therefore, a heuristic approach must be taken, identifying hyperplane candidates and calculating a metric to choose the 'best' plane.

Candidate hyperplanes

In order for a plane to be the next candidate node, it must be unbounded in all dimensions but one.

Decision space \rightarrow decision tree

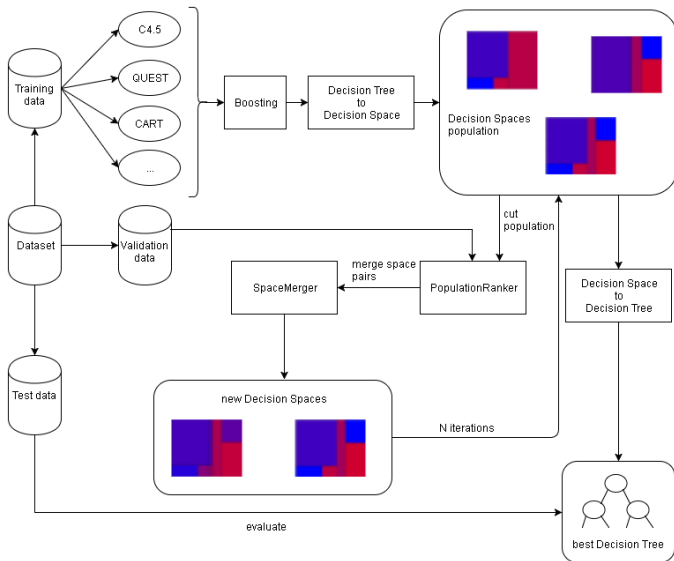


Decision space \rightarrow decision tree

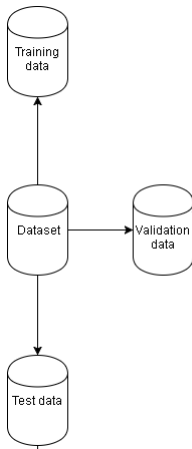
Finding 'best' candidate hyperplane

Apply metric function to each plane, these include:

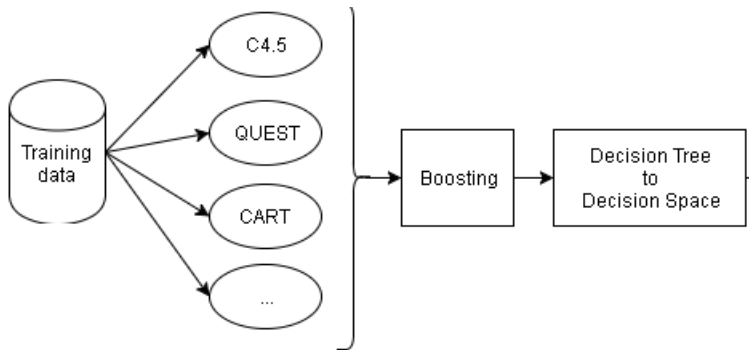
- ▶ information gain and Gini from C4.5 and CART respectively
- ▶ pick plane from most correlated feature (χ^2 - and ANOVA F -test from QUEST)
- ▶ pick plane that divide space in two most equal subspaces (using surface/volume or counting number of planes)
- ▶ combination



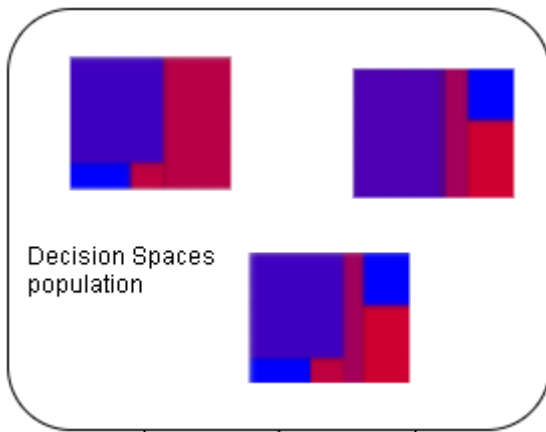
Splitting the data



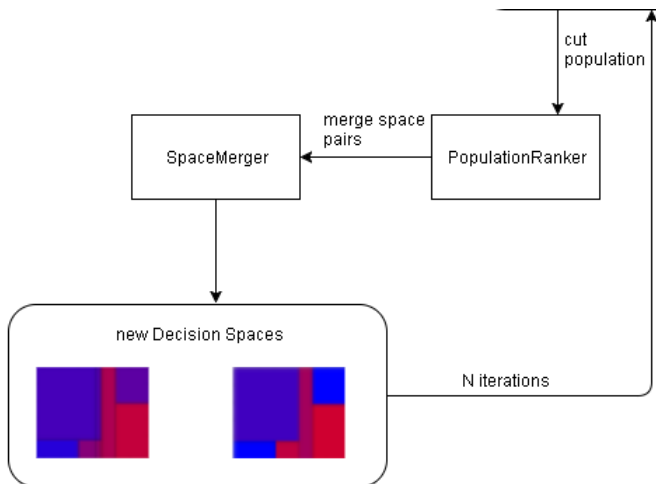
Generate different decision trees



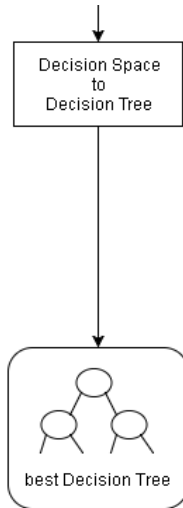
Generate different decision trees



Genetic merging



Final iteration



Evaluating our algorithm

5 datasets from UCI
optimal parameters, feature selection when needed and k-fold
cross-validation

Heart disease dataset



- ▶ medical dataset
- ▶ 270 samples
- ▶ 13 features: 6 continuous (max heartrate), 7 discrete (sex)
- ▶ 2 classes: sick or healthy
- ▶ more healthy than sick samples
- ▶ false negatives can cost lives!

Car evaluation dataset



- ▶ 1725 samples
- ▶ only 6 features, all categories (price, doors, ...)
- ▶ 4 classes: unacceptable, acceptable, good and very good
- ▶ VERY imbalanced (70% unacc, 22.2% acc)

Iris flowers dataset



- ▶ very small dimensions
- ▶ 150 flowers
- ▶ 4 continuous features (petal width and sepal length)
- ▶ 3 classes: setosa, versicolor, virginica
- ▶ perfectly balanced (50-50-50)

Shuttle classification dataset



- ▶ originally: 58000 samples corresponding to aircrafts
- ▶ here: 14500 samples
- ▶ 9 numerical attributes
- ▶ 7 different types of shuttles: 2 of them contained less than 10 samples (discarded)

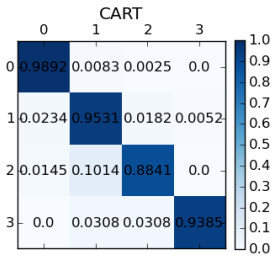
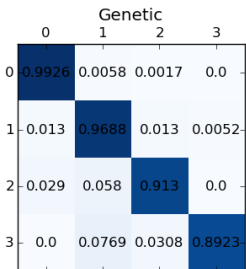
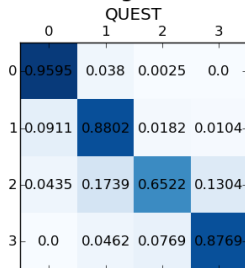
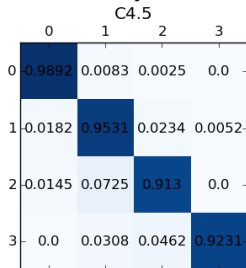
Nursery application dataset

- ▶ 12960 samples (applications for nursery school)
- ▶ 8 categorical features
- ▶ 5 classes: not recommended, recommended, very recommended, priority and special priority
- ▶ only 2 samples were recommended (discarded)

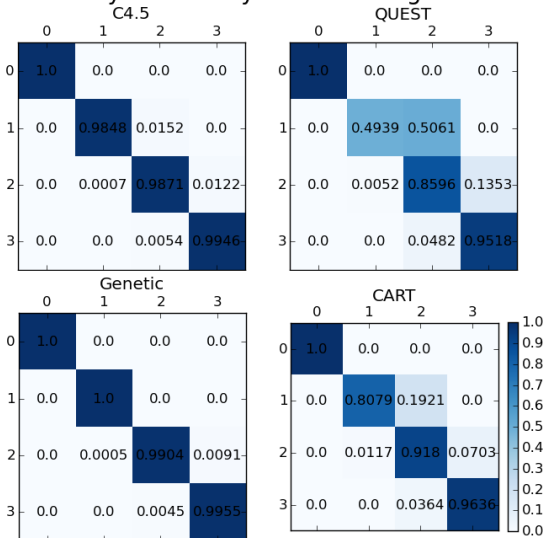


Dataset	Folds	C4.5	CART	QUEST	Genetic
Heart disease	5	<u>0.8067</u>	0.7844	0.7844	<u>0.8067</u>
	10	<u>0.8104</u>	0.7732	0.7881	0.7993
Iris	3	0.9533	0.9467	0.9467	<u>0.96</u>
	5	0.9467	0.9333	0.9467	<u>0.9533</u>
Cars	3	<u>0.9722</u>	0.9693	0.9229	0.9693
	5	0.9711	0.9682	0.9241	<u>0.9786</u>
	10	0.9756	0.9751	0.9265	<u>0.9803</u>
Shuttle	3	0.9987	0.9983	0.9964	<u>0.9988</u>
	5	0.9986	0.9981	0.9962	<u>0.9988</u>
	10	0.9990	0.9987	0.9941	<u>0.9992</u>
Nursery	3	0.9890	0.9431	0.9147	<u>0.9914</u>
	5	0.9918	0.9498	0.9251	<u>0.9958</u>
	10	0.9937	0.9568	0.9259	<u>0.9954</u>

Accuracy on cars dataset using 10 folds



Accuracy on nursery dataset using 10 folds



Intro

Machine learning - DT's

Platform requirements

Genetic merging of DT's

Mobile application

Visualization

Backend and data exposure

Conclusion & future work

Intro

Machine learning - DT's

Platform requirements

Genetic merging of DT's

Mobile application

Visualization

Backend and data exposure

Conclusion & future work

Bedankt

Bedank voor uw aandacht

No written word,
No spoken plea,
Can teach the youth what they should be,
Nor all the books on all the shelves,
It's what the teachers are themselves

Intro

Platform requirements

Mobile application

Backend and data exposure

Machine learning - DT's

Genetic merging of DT's

Visualization

Conclusion & future work