## Deadlines

<table>
<thead>
<tr>
<th>Assignment</th>
<th>Date of assignment</th>
<th>Deadline (midnight 23:59)</th>
</tr>
</thead>
<tbody>
<tr>
<td>HW1</td>
<td>Sep 5</td>
<td>Sep 18</td>
</tr>
<tr>
<td>HW2</td>
<td>Sep 19</td>
<td>Oct 2</td>
</tr>
<tr>
<td>HW3</td>
<td>Oct 3</td>
<td>Oct 16</td>
</tr>
<tr>
<td>Paper summary</td>
<td>Oct 10</td>
<td>Oct 23</td>
</tr>
<tr>
<td>HW4</td>
<td>Oct 17</td>
<td>Oct 30</td>
</tr>
<tr>
<td>HW5</td>
<td>Oct 31</td>
<td>Nov 13</td>
</tr>
<tr>
<td>HW6</td>
<td>Nov 21</td>
<td>Dec 4</td>
</tr>
<tr>
<td>Project</td>
<td>Oct 14</td>
<td>Dec 12 - 14</td>
</tr>
</tbody>
</table>

*All deadlines are subject to change, check out Slack and website for updates*
“In previous episodes...”
There are two main ways of looking at this (A and B):

**way A:** something wrong with a model

**way B:** something wrong with a data

**Epoch #5**

Overfitting
Regularisation methods

Way A methods

Way B methods
Way A methods
Explicit regularisation methods
(Dropout, L1/L2 regularisation)
Despite some differences, these methods belong to so called **explicit regularisation methods**

They **explicitly limit** predictive power of the model by introducing **additional constraints**

**L2/L1 regularisation**  VS  **Dropout**
Way B methods
Implicit regularisation methods
These are the most basic transforms
More advanced things are available via github/aleju/imgaug
Explicit regularisation methods
(Dropout, L1/L2 regularisation)

Implicit regularisation methods
There are two main ways of looking at this (A and B):

**way A:** something wrong with a model

**way B:** something wrong with a data

**Epoch #5**

There is a graph showing loss over epochs, indicating overfitting.
There are plenty of other cool ways to get **better performance** and **fight overfitting**!
There are plenty of other cool ways to get better performance and fight overfitting!

Weak learners, ENSEMBLE!
Supervised Learning
Unsupervised Learning
Performance metrics
Deep Learning
Ensemble learning
Other

Overview of ML
Course organisation
Hierarchical clustering
K-means
T-SNE
UNAP
PCA
Model implementation
Model selection
Algorithm
Model
Data preprocessing
Selection
Preprocessing
Linear regression
Decision trees
Overfitting
Tran/val split
Cross Validation
Algorithm
Classification vs regression
Accuracy
MSE & RMSE
Recall, precision
F1-score
Confusion matrix
ROC & AUC
Vanishing Gradients
Convolutional NNs

Artificial neuron
Forward path
Activation Functions
Gradient descent
Backpropagation algorithm
Vanishing Gradients
Convolutional NNs

Backpropagation algorithm
Vanishing Gradients
Convolutional NNs

Regularisation
L1 & L2 regularisation
Training loop

Dropout
Weight decay
Ridge regression
Lasso regression

Data augmentation
Basic ensembling

Vanishing Gradients
Convolutional NNs

Guest talk
Concluding remarks
Final presentations
HW1

Six homeworks
(10 points each)

Paper review
(15 points)

Now you are ready for real-life tasks
Now you are ready for real-life tasks
James Surowiecki, *The Wisdom of Crowds*
In 1907, 787 villagers tried to guess the weight of ox.
In 1907, 787 villagers tried to guess the weight of ox James Surowiecki, *The Wisdom of Crowds*

None of them guessed it correctly, but the average guess (542.9 kg) was very close to actual weight of ox (543.4 kg)

James Surowiecki, *The Wisdom of Crowds*
Test
Multiple models are built on training data

Basic ensemble
Multiple models are built on training data

Similar idea can be applied for **classification**

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>LR</td>
<td>3.18</td>
<td>5.55</td>
</tr>
<tr>
<td>Ridge</td>
<td>3.23</td>
<td>5.17</td>
</tr>
<tr>
<td>Lasso</td>
<td>3.57</td>
<td>4</td>
</tr>
<tr>
<td>Average</td>
<td>3.32</td>
<td>4.89</td>
</tr>
<tr>
<td>TRUE</td>
<td>4</td>
<td>5</td>
</tr>
</tbody>
</table>
How do we know who of these people will be successful in running the country?
Majority voting

<table>
<thead>
<tr>
<th>Test</th>
<th>Model 1</th>
<th>Model 2</th>
<th>Model 3</th>
<th>Ensemble</th>
<th>TRUE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Dog</td>
<td>Dog</td>
<td>Cat</td>
<td>Dog</td>
<td>Dog</td>
</tr>
</tbody>
</table>

Training data

- M1
- M2
- M3

- Model 1: Dog
- Model 2: Dog
- Model 3: Cat
- Ensemble: Dog
- TRUE: Dog
Hard voting

Training data

<table>
<thead>
<tr>
<th>Test</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Model 1</td>
<td>Dog</td>
</tr>
<tr>
<td>Model 2</td>
<td>Dog</td>
</tr>
<tr>
<td>Model 3</td>
<td>Cat</td>
</tr>
<tr>
<td>Ensemble</td>
<td>Dog</td>
</tr>
<tr>
<td><strong>TRUE</strong></td>
<td>Dog</td>
</tr>
</tbody>
</table>
M3

M2

Hard voting

Model 1: Dog
Model 2: Dog
Model 3: Cat
Ensemble: Dog
TRUE: Dog

Soft voting

Test

Model 1: Dog
Model 2: Dog
Model 3: Cat
Ensemble: Dog
TRUE: Dog
Hard voting

<table>
<thead>
<tr>
<th>Test</th>
<th>Dog</th>
<th>Cat</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model 1</td>
<td>Dog</td>
<td>0.8</td>
</tr>
<tr>
<td>Model 2</td>
<td>Dog</td>
<td>0.6</td>
</tr>
<tr>
<td>Model 3</td>
<td>Cat</td>
<td>0.4</td>
</tr>
<tr>
<td>Ensemble</td>
<td>Dog</td>
<td>1.0</td>
</tr>
</tbody>
</table>

Soft voting
Hard voting

- Model 1: Dog, 0.8
- Model 2: Dog, 0.6
- Model 3: Cat, 0.4
- Ensemble: Dog, 0.6

Soft voting

- Model 1: Dog, 0.8
- Model 2: Dog, 0.6
- Model 3: Cat, 0.4
- Ensemble: Dog, 0.6

TRUE: Dog, 1.0
What does such voting assume about the models?
Majority voting assumes equal weight of each model’s vote.

<table>
<thead>
<tr>
<th></th>
<th>Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model 1</td>
<td>Dog</td>
</tr>
<tr>
<td>Model 2</td>
<td>Dog</td>
</tr>
<tr>
<td>Model 3</td>
<td>Cat</td>
</tr>
<tr>
<td>Ensemble</td>
<td>Dog</td>
</tr>
<tr>
<td>TRUE</td>
<td>Dog</td>
</tr>
</tbody>
</table>

Hard voting
Majority voting assumes **equal weight** of each model's vote.
Does this always make sense?

Majority voting assumes equal weight of each model’s vote.
Let’s assume that we trust one model more than the other model.

<table>
<thead>
<tr>
<th>Test</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model 1</td>
<td>Dog</td>
</tr>
<tr>
<td>Model 2</td>
<td>Dog</td>
</tr>
<tr>
<td>Model 3</td>
<td>Cat</td>
</tr>
<tr>
<td>Ensemble</td>
<td></td>
</tr>
<tr>
<td>TRUE</td>
<td>Dog</td>
</tr>
</tbody>
</table>
### Hard Voting

<table>
<thead>
<tr>
<th>Test</th>
<th>Weight</th>
<th>Votes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model 1</td>
<td>Dog</td>
<td>0.33</td>
</tr>
<tr>
<td>Model 2</td>
<td>Dog</td>
<td>0.37</td>
</tr>
<tr>
<td>Model 3</td>
<td>Cat</td>
<td>0.30</td>
</tr>
<tr>
<td>Ensemble</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TRUE</td>
<td>Dog</td>
<td></td>
</tr>
</tbody>
</table>

The ensemble voting method decides on a `Dog` classification as it received the highest number of votes with a weight of 0.37.
<table>
<thead>
<tr>
<th>Test</th>
<th>Weight</th>
<th>Votes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model 1</td>
<td>Dog</td>
<td>0.33</td>
</tr>
<tr>
<td>Model 2</td>
<td>Dog</td>
<td>0.37</td>
</tr>
<tr>
<td>Model 3</td>
<td>Cat</td>
<td>0.30</td>
</tr>
<tr>
<td>Ensemble</td>
<td>70 x Dog vs 30 x Cat</td>
<td></td>
</tr>
<tr>
<td>TRUE</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Hard voting**
Things would change if there would be 3 classes (dog, cat, rabbit) instead of 2.

<table>
<thead>
<tr>
<th>Test</th>
<th>Weight</th>
<th>Votes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model 1</td>
<td>Rat</td>
<td>0.33</td>
</tr>
<tr>
<td>Model 2</td>
<td>Dog</td>
<td>0.37</td>
</tr>
<tr>
<td>Model 3</td>
<td>Cat</td>
<td>0.30</td>
</tr>
<tr>
<td>Ensemble</td>
<td>TRUE</td>
<td></td>
</tr>
</tbody>
</table>

Hard voting
Things would change if there would be 3 classes (dog, cat, rabbit) instead of 2.
How weights work for **soft voting**?

<table>
<thead>
<tr>
<th></th>
<th>Dog</th>
<th>Cat</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model 1</td>
<td>0.8</td>
<td>0.2</td>
<td>0.33</td>
</tr>
<tr>
<td>Model 2</td>
<td>0.6</td>
<td>0.4</td>
<td><strong>0.37</strong></td>
</tr>
<tr>
<td>Model 3</td>
<td>0.4</td>
<td>0.6</td>
<td>0.30</td>
</tr>
<tr>
<td>Ensemble</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>TRUE</strong></td>
<td>1.0</td>
<td>0.0</td>
<td></td>
</tr>
</tbody>
</table>

**Soft voting**
How weights work for **soft voting**?

<table>
<thead>
<tr>
<th></th>
<th>Dog</th>
<th>Cat</th>
<th>Weight</th>
<th>W. Dog</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model 1</td>
<td>0.8</td>
<td>0.2</td>
<td>0.33</td>
<td>0.8x0.33</td>
</tr>
<tr>
<td>Model 2</td>
<td>0.6</td>
<td>0.4</td>
<td><strong>0.37</strong></td>
<td>0.6x0.37</td>
</tr>
<tr>
<td>Model 3</td>
<td>0.4</td>
<td>0.6</td>
<td>0.30</td>
<td>0.4x0.30</td>
</tr>
<tr>
<td>Ensemble</td>
<td><strong>1.0</strong></td>
<td>0.0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Soft voting**
How weights work for **soft voting**?

<table>
<thead>
<tr>
<th></th>
<th>Dog</th>
<th>Cat</th>
<th>Weight</th>
<th>W. Dog</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model 1</td>
<td>0.8</td>
<td>0.2</td>
<td>0.33</td>
<td>0.264</td>
</tr>
<tr>
<td>Model 2</td>
<td>0.6</td>
<td>0.4</td>
<td><strong>0.37</strong></td>
<td><strong>0.222</strong></td>
</tr>
<tr>
<td>Model 3</td>
<td>0.4</td>
<td>0.6</td>
<td>0.30</td>
<td>0.120</td>
</tr>
<tr>
<td>Ensemble</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>TRUE</strong></td>
<td>1.0</td>
<td>0.0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Soft voting**
How weights work for **soft voting**?

<table>
<thead>
<tr>
<th></th>
<th>Dog</th>
<th>Cat</th>
<th>Weight</th>
<th>W.Dog</th>
<th>W.Cat</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model 1</td>
<td>0.8</td>
<td>0.2</td>
<td>0.33</td>
<td>0.264</td>
<td>0.066</td>
</tr>
<tr>
<td>Model 2</td>
<td>0.6</td>
<td>0.4</td>
<td><strong>0.37</strong></td>
<td><strong>0.222</strong></td>
<td>0.148</td>
</tr>
<tr>
<td>Model 3</td>
<td>0.4</td>
<td>0.6</td>
<td>0.30</td>
<td>0.120</td>
<td>0.18</td>
</tr>
<tr>
<td>Ensemble</td>
<td>0.6</td>
<td>0.4</td>
<td>0.606</td>
<td><strong>0.394</strong></td>
<td></td>
</tr>
<tr>
<td><strong>TRUE</strong></td>
<td>1.0</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Soft voting**
How weights work for **soft voting**?

<table>
<thead>
<tr>
<th></th>
<th>Dog</th>
<th>Cat</th>
<th>Weight</th>
<th>W.Dog</th>
<th>W.Cat</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model 1</td>
<td>0.8</td>
<td>0.2</td>
<td>0.15</td>
<td>0.12</td>
<td>0.03</td>
</tr>
<tr>
<td>Model 2</td>
<td>0.6</td>
<td>0.4</td>
<td>0.15</td>
<td>0.09</td>
<td>0.06</td>
</tr>
<tr>
<td>Model 3</td>
<td>0.4</td>
<td>0.6</td>
<td>0.70</td>
<td>0.28</td>
<td>0.42</td>
</tr>
<tr>
<td>Ensemble</td>
<td>0.6</td>
<td>0.4</td>
<td>0.49</td>
<td>0.51</td>
<td></td>
</tr>
<tr>
<td><strong>TRUE</strong></td>
<td>1.0</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Soft voting**
If you know that your models are not equally useful, **how do you choose weights?**
We can estimate the weight of each model based on \textbf{CV} on training data.

If you know that your models are not equally useful, \textbf{how do you choose weights?}
Ensembles assume models to have uncorrelated errors.
Ensembles assume models to have **uncorrelated errors**.
Ensembles assume models to have **uncorrelated errors**.

M1  M2  M3
Ensembles assume models to have uncorrelated errors.
How to achieve **uncorrelated errors** of models with **only one type** of algorithm (e.g. decision tree)?

Training data
How to achieve uncorrelated errors of models with only one type of algorithm (e.g. decision tree)?

Training data

We can play around with the data!
Bootstrapping

Training data

Bootstrapped data

Random 70%

Training on different parts of data produces diverse models with uncorrelated errors
Bootstrapping

Training data

Bootstrapped data

Random 70%
Bootstrapping

Training data → Bootstrapped data

Random 70%

Validation data
Bootstrapping
Bootstrapping

Training data

Bootstrapped data

Validation data

Random 70%

Good old majority vote

Dog

Cat
Bootstrapping

Training data

Bootstrapped data

Validation data

Random 70%

Dog

Cat

Dog
Bootstrapping

Training data

Bootstrapped data

Validation data

Random 70%

Dog

Cat

Aggregation (majority vote or averaging)
Bootstrapping + Aggregation = Bagging
Bootstrapping + Aggregation = Bagging

Training data

Validation data

Bootstrapped data

Random 70%
Bootstrapping + Aggregation = Bagging

Bootstrapped data

Training data

Random 70%

Validation data
Decision Tree Algorithm

By asking a simple question about value of independent variable it tries to predict a value of dependent variable.
Decision Tree Algorithm

By asking a simple question about value of $X_1$ and $X_2$ it tries to predict a class (1, 0)
Decision Tree Algorithm

By asking a simple question about value of $X_1$ and $X_2$ it tries to predict a class $(1,0)$.
Decision Tree Algorithm

By asking a simple **question** about value of $X_1$ and $X_2$, it tries to predict a class $(1, 0)$.
Decision Tree Algorithm

By asking a simple question about value of $X_1$ and $X_2$ it tries to predict a class $(1, 0)$
Let’s apply **bagging** to learn **three different** decision trees.
Let’s apply **bagging** to learn **three different** decision trees.
Let’s apply **bagging** to learn **three different** decision trees.
Let’s apply **bagging** to learn **three different** decision trees.
Building the **first** decision tree based on data in the **bag #1**
What we need to do in order to build a decision tree?

Building the **first** decision tree based on data in the **bag #1**
Building the **first** decision tree based on data in the **bag #1**

1. Need to **evaluate** all possible splits
Building the **first** decision tree based on data in the bag #1

1. Need to **evaluate** all possible splits

What are the **all possible splits**
Building the **first** decision tree based on data in the **bag #1**

1. Need to **evaluate** all possible splits

What are the **all possible splits**
Building the **first** decision tree based on data in the **bag #1**

1. Need to **evaluate** all possible splits

What are the **all possible splits**

Are these **are only possible splits**?
Building the first decision tree based on data in the bag #1

X_1 and X_2 are both features that we can use to divide data.

1. Need to evaluate all possible splits.

What are the all possible splits?

Are these are only possible splits?
Building the **first** decision tree based on data in the bag #1

**X**\(_1\) and **X**\(_2\) are both features that we can use to divide data.

1. Need to **evaluate** all possible splits

What are the **all possible splits**?

Are these **are only possible splits**?
Building the **first** decision tree based on data in the bag #1

1. Need to **evaluate** all possible splits

What are the **all possible splits**?

Are **all possible splits** are also **reasonable** splits?
Building the **first** decision tree based on data in the bag #1

1. Need to evaluate all possible splits

What are the all possible splits

These splits are **not** reasonable (data is not divided)

Are **all possible splits** are also **reasonable** splits?
Building the **first** decision tree based on data in the bag #1

1. Need to **evaluate** all possible splits

What are the all possible splits
Building the **first** decision tree based on data in the **bag #1**

1. Need to **evaluate** all possible splits

What are the **all possible splits** (but also **reasonable**)?
Building the **first** decision tree based on data in the **bag #1**

How do we **evaluate** splits?

1. Need to **evaluate** all possible splits

What are the all possible splits (but also reasonable)?
Building the first decision tree based on data in the bag #1

How do we evaluate splits?

1. Need to evaluate all possible splits

What are the all possible splits (but also reasonable)?

We can compute accuracy of the split
Building the **first** decision tree based on data in the bag #1

How do we **evaluate** splits?

1. Need to **evaluate** all possible splits
   - What are the **all possible splits** (but also **reasonable**)?

We can compute **accuracy** of the split
Building the first decision tree based on data in the bag #1

**Accuracy** = (# of guessed correctly)/(# of all)

We can compute accuracy of the split

How do we evaluate splits?

1. Need to evaluate all possible splits

What are the all possible splits (but also reasonable)?
Building the first decision tree based on data in the bag #1

Accuracy = (# of guessed correctly)/(# of all)

How do we evaluate splits?

1. Need to evaluate all possible splits (but also reasonable).

We can compute accuracy of the split.
Building the first decision tree based on data in the bag #1.

We can compute accuracy of the split:

\[ \text{Accuracy} = \frac{3}{\text{(# of all)}} \]

How do we evaluate splits?

1. Need to evaluate all possible splits (but also reasonable).

What are the all possible splits (but also reasonable)?
Building the first decision tree based on data in the bag #1

Accuracy = $\frac{3}{4}$

We can compute accuracy of the split

How do we evaluate splits?

1. Need to evaluate all possible splits (but also reasonable)?

What are the all possible splits (but also reasonable)?
Building the **first** decision tree based on data in the bag #1

**Accuracy** = 75%

We can compute **accuracy** of the split

How do we **evaluate** splits?

1. Need to **evaluate** all possible splits
2. What are the all possible splits (but also **reasonable**)?
Building the **first** decision tree based on data in the bag #1

We can compute **accuracy** of the split

How do we **evaluate** splits?

1. Need to **evaluate** all possible splits (but also **reasonable**)?
This split is **double sided**
This split is **double sided**.
This split is **double sided**

What is the **accuracy** of the mirrored split?
This split is **double sided**

What is the **accuracy** of the mirrored split?
This split is **double sided**

Let’s differentiate them by the **arrow**
Building the **first** decision tree based on data in the bag #1

We can compute **accuracy** of the split

How do we **evaluate** splits?

1. Need to **evaluate** all possible splits (but also **reasonable**)?

What are the **all possible splits** (but also **reasonable**)?
Building the first decision tree based on data in the bag #1

How do we evaluate splits?

1. Need to evaluate all possible splits (but also reasonable)?

We can compute accuracy of the split
Building the first decision tree based on data in the bag #1

We can compute accuracy of the split

It seems a lot more splits than we thought at first…

1. Need to evaluate all possible splits (but also reasonable)?

We can compute accuracy of the split
1. Need to evaluate all possible splits.

How do we evaluate splits?

What are the all possible splits (but also reasonable)?

Building the first decision tree based on data in the bag #1

We can compute accuracy of the split.

It seems a lot more splits than we thought at first…

Worry not, accuracy of the split defines the accuracy of the reverse split.

We can compute accuracy of the split.
Building the **first** decision tree based on data in the bag #1

We can compute **accuracy** of the split

1. Need to evaluate all possible splits

What are the all possible splits (but also reasonable)?

How do we **evaluate** splits?
Building the first decision tree based on data in the bag #1

We can compute accuracy of the split

How do we evaluate splits?

1. Need to evaluate all possible splits
   What are the all possible splits (but also reasonable)?
Building the **first** decision tree based on data in the bag #1

We can compute **accuracy** of the split

How do we evaluate splits?

1. Need to evaluate all possible splits

What are the all possible splits (but also reasonable)?
Building the **first** decision tree based on data in the bag #1

We can compute **accuracy** of the split

1. Need to evaluate all possible splits
   - What are the all possible splits (but also **reasonable**)?

   How do we **evaluate** splits?
Building the **first** decision tree based on data in the bag #1

We can compute **accuracy** of the split.

How do we evaluate splits?

1. Need to evaluate all possible splits

What are the all possible splits (but also reasonable)?
Building the **first** decision tree based on data in the bag #1

We can compute **accuracy** of the split

How do we **evaluate** splits?

1. Need to **evaluate** all possible splits
   (but also **reasonable**)?

What are the **all possible splits**?
Building the **first** decision tree based on data in the bag #1

We can compute **accuracy** of the split

How do we **evaluate** splits?

1. Need to **evaluate** all possible splits (but also **reasonable**)?

What are the **all possible splits**?
Building the **first** decision tree based on data in the bag #1

**We can compute** accuracy of the split

**How do we evaluate** splits?

1. Need to **evaluate** all possible splits (but also **reasonable**)?
Building the **first** decision tree based on data in the bag #1

We can compute **accuracy** of the split

How do we **evaluate** splits?

1. Need to **evaluate** all possible splits

What are the all possible splits (but also **reasonable**)?
Building the *first* decision tree based on data in the bag #1.

We can compute **accuracy** of the split.

How do we **evaluate** splits?

1. Need to **evaluate** all possible splits.
   What are the **all possible splits** (but also **reasonable**)?
Building the **first** decision tree based on data in the bag #1

How do we **evaluate** splits?

1. Need to **evaluate** all possible splits

What are the **all possible splits** (but also **reasonable**)?

We can compute **accuracy** of the split
Building the first decision tree based on data in the bag #1

How do we evaluate splits?

1. Need to evaluate all possible splits

What are the all possible splits (but also reasonable)?

We can compute accuracy of the split
Building the first decision tree based on data in the bag #1

How do we evaluate splits?

1. Need to evaluate all possible splits (but also reasonable)?

We can compute accuracy of the split
Building the **first** decision tree based on data in the bag #1

How do we **evaluate** splits?

1. Need to **evaluate** all possible splits
   (but also **reasonable**)?
Building the **first** decision tree based on data in the bag #1

1. Need to **evaluate** all possible splits
Building the first decision tree based on data in the bag #1

1. Need to evaluate all possible splits

2. Choose the best split
Building the **first** decision tree based on data in the bag #1

1. Need to **evaluate** all possible splits

2. Choose the **best** split
Building the first decision tree based on data in the bag #1

1. Need to evaluate all possible splits

2. Choose the best split

Which splits are we interested in?
Building the **first** decision tree based on data in the bag #1

1. Need to **evaluate** all possible splits

2. Choose the **best** split

Which splits are we interested in?
Building the **first** decision tree based on data in the bag #1

1. Need to evaluate all possible splits

2. Choose the **best** split

**Which splits** are we interested in?
Building the **first** decision tree based on data in the bag #1

1. Need to **evaluate** all possible splits

2. Choose the **best** split

**Which splits** are we interested in?
Building the first decision tree based on data in the bag #1

1. Need to **evaluate** all possible splits
2. Choose the **best** split
Building the **first** decision tree based on data in the bag #1

Here is our tree so far:

\[ X_1 > 4.5 \]

- **False**
  - Class = 1

- **True**
  - Class = 0
Are we **done** with a tree for **bag #1**?

Here is our tree so far:

- **$X_1 > 4.5$**
  - **False** → **Class = 1**
  - **True** → **Class = 0**
Are we **done** with a tree for **bag #1**?

Here is our tree so far:

- **$X_1 > 4.5$**
  - **False**
    - Class = 1
      - 66.6% correct
  - **True**
    - Class = 0
      - 100% correct
Are we **done** with a tree for **bag #1**?

Here is our tree so far:

- **Class = 1**
- **Class = 0**

Can we **improve** accuracy of this leaf?
Let’s add another layer
Let’s add another layer

1. Need to evaluate all possible splits

2. Choose the best split
Let’s add another layer

1. Need to **evaluate** all possible splits

2. Choose the **best** split
Let’s add another layer

1. Need to **evaluate** all possible splits
2. Choose the **best** split
Let’s add another layer

1. Need to evaluate all possible splits

2. Choose the best split
Let’s add another layer

1. Need to evaluate all possible splits

2. Choose the best split
Let’s add another layer

1. Need to **evaluate** all possible splits

2. Choose the **best** split
The final decision tree for **bag #1**

- For $X_1 = 2.5$:
  - $X_2 = 4.5$, 66%
  - $X_2 = 2.5$

- For $X_1 = 4.5$:
  - $X_2 = 5.5$, 66%
The final decision tree for bag #1
As we want to build three trees we need three independent **bootstrap**s.
As we want to build three trees we need three independent bootstrapts.
As we want to build three trees we need three independent bootstraps.
As we want to build three trees we need three independent bootstrap samples.
As we want to build three trees we need three independent bootstrapts.
As we want to build three trees we need three independent bootstraps.
As we want to build three trees we need three independent bootstraps.
As we want to build three trees we need three independent bootstraps.
As we want to build three trees we need three independent bootstrapts.
As we want to build three trees we need three independent bootstrap samples.
We can **build** the **ensemble** on the original data by **combining** trees.
We can build the ensemble on the original data by combining trees.
Let’s add a very simple **twist** to this algorithm.
As last time we shall start with generating bootstraps.
As last time we shall start with generating bootstraps.
Something will change with respect to **how we build trees**
Something will change with respect to **how we build trees**

Hint: we will have to **toss coins**, again :)}
Familiar algorithm of building tree had **2 steps**:
Familiar algorithm of building tree had **2 steps**: 

1. Need to evaluate all possible splits

2. Choose the **best split**
Here we add **a new step:**

1. Need to evaluate all possible splits
2. Choose the **best** split
Here we add **a new step**:

0. Choose a **random set** of features

1. Need to evaluate all possible splits

2. Choose the **best split**
Here we add a new step:

0. Choose a random set of features

1. Need to evaluate all possible splits

2. Choose the best split

Here we have only 2 features ($X_1$ and $X_2$), so we will choose one, but normally you would keep about 80% of the original features.
Here we add a new step:

0. Choose a random set of features

1. Need to evaluate all possible splits

2. Choose the best split

Heads, means we keep $X_2$

Tail, means we keep $X_1$
Here we add a new step:

0. Choose a **random set** of features

1. Need to evaluate all possible splits

2. Choose the **best** split
Here we add a new step:

0. Choose a random set of features

1. Need to evaluate all possible splits

2. Choose the best split
Here we add a new step:

0. Choose a **random set** of features

1. Need to evaluate all possible splits

2. Choose the **best** split

We keep $X_2$
Here we add a new step:

0. Choose a random set of features

1. Need to evaluate all possible splits

2. Choose the best split

We keep $X_2$ and get rid of $X_1$
Here we add a new step:

0. Choose a random set of features

1. Need to evaluate all possible splits

2. Choose the best split

We keep $X_2$ and get rid of $X_1$
0. Choose a random set of features

1. Need to evaluate all possible splits

2. Choose the best split
0. Choose a **random set** of features

1. Need to evaluate **all possible splits**

2. Choose the **best split**
0. Choose a \textit{random set} of features

1. Need to evaluate \textit{all possible splits}

2. Choose the \textit{best split}

Thanks to \textbf{step 0} we have only one option left…
0. Choose a random set of features

1. Need to evaluate all possible splits

2. Choose the best split
0. Choose a **random set** of features

1. Need to evaluate all possible splits

2. Choose the **best split**

Both *original* and *reverse* are equally bad, so we **toss a coin** again…
0. Choose a random set of features

1. Need to evaluate all possible splits

2. Choose the best split
This is what we have built so far:

\[ X_2 > 4.5 \]

- True: Class = 0
- False: Class = 1

Bag #1
This is what we have built so far:

In principle you may go **deeper** into each one of the leaves.

- **Class = 1**
- **Class = 0**
This is what we have built so far:

In principle you may go **deeper** into each one of the leaves.

Every time you would need to **toss a coin** to select a **new random set** of features from initial features.
We kept $X_2$
We kept $X_2$
We kept $X_2$.

We kept $X_1$.
We kept $X_2$.
We kept $X_2$.

We keep $X_1$. 

80% 

80% 

80%
We kept $X_2$

We keep $X_1$
We kept $X_2$
We can build an ensemble
We can build an ensemble
We can build an ensemble.
This model is imperfect
This model is **imperfect**

We used only **1 layer** trees

66%
This model is imperfect.

We used only 1 layer of trees.

Selecting features randomly for each node, has been shown to produce great results in practice.
This model is **imperfect**

We used only **1 layer** trees

Selecting features randomly for each node, has been shown to **produce great results** in practice (acts as **regularisation**).
The Random Forest algorithm
The **Random Forest** algorithm

Form several **bags** (using bootstrapping)

For each **bag** build a **tree**

For each **node** in the **tree** choose a **random set of features**

Merge predictions using **majority vote** or **averaging**
The **Random Forest** algorithm

Form several **bags** (using bootstrapping)

For each bag, build a tree

For each node in the tree, choose a random set of features

Merge predictions using **majority vote** or averaging

Plenty of **coin tossing**!
Basic ensembling

Weighted ensembling

Bagging ensembling

The Random Forest algorithm

Form several bags (using bootstrapping)
For each bag build a tree
For each node in the tree choose random set of features
Merge predictions using majority vote or averaging

Random Forest

Next time!

Stacking

Blending
That's all Folks!