UCLA Health | Jonsson Comprehensive Cancer Center

BOUTROS LAB

Random Forests

11-21-2024

Nick Nunley

Outline

- Shannon Entropy
- Decision trees
- Random forests

Shannon Entropy

Mathematical definition

$$H(X) = -\sum_{i} P(X)_i log_2(P(X)_i)$$

i represents a possible outcome out of all possible outcomes for a random process

Mathematical definition

$$H(X) = -\sum_{i} \underbrace{P(X)_i}_{\text{Probability of outcome}} \underbrace{log_2(P(X)_i)}_{\text{Information content of outcome}}$$

i represents a possible outcome out of all possible outcomes for a random process

Shannon Entropy example

$$H_{Frag} = -\frac{16}{16}\ln\left(\frac{16}{16}\right) = 0$$

$$H_{Frag} = -\frac{8}{16}\ln\left(\frac{8}{16}\right) = 0.347$$

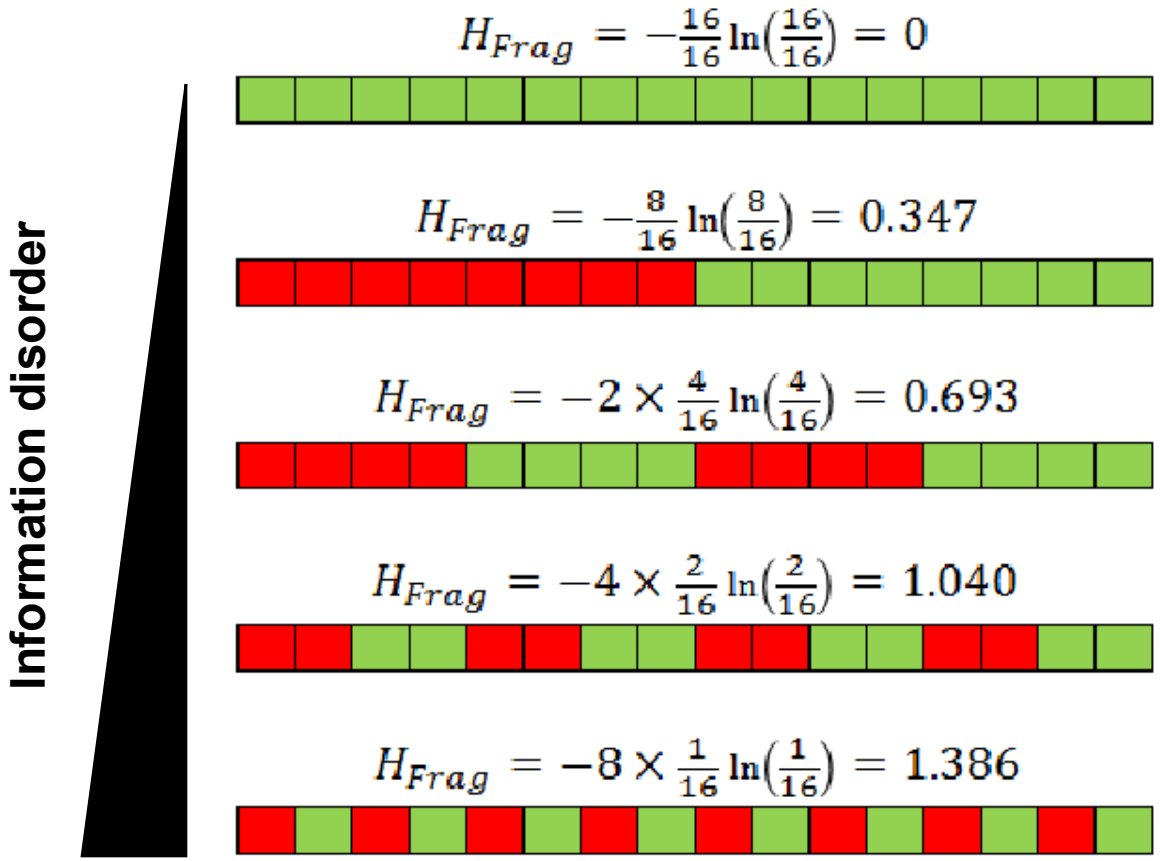
$$H_{Frag} = -2 \times \frac{4}{16}\ln\left(\frac{4}{16}\right) = 0.693$$

$$H_{Frag} = -4 \times \frac{2}{16}\ln\left(\frac{2}{16}\right) = 1.040$$

$$H_{Frag} = -8 \times \frac{1}{16}\ln\left(\frac{1}{16}\right) = 1.386$$

https://www.researchgate.net/publication/261042470_Simulation_Results_of_Shannon_Entropy_based_Flexgrid_Routing_and_Spectrum_Assignment_on_a_Real_Network_Topology

Shannon Entropy example



https://www.researchgate.net/publication/261042470_Simulation_Results_of_Shannon_Entropy_based_Flexgrid_Routing_and_Spectrum_Assignment_on_a_Real_Network_Topology

Decision trees

Decision tree definition

- A supervised learning approach represented with a flowchart-like tree data structure used to make decisions or predictions

- **Internal nodes** represent conditionals for evaluating/predicting the target outcome and **leaf nodes** represent the target outcome

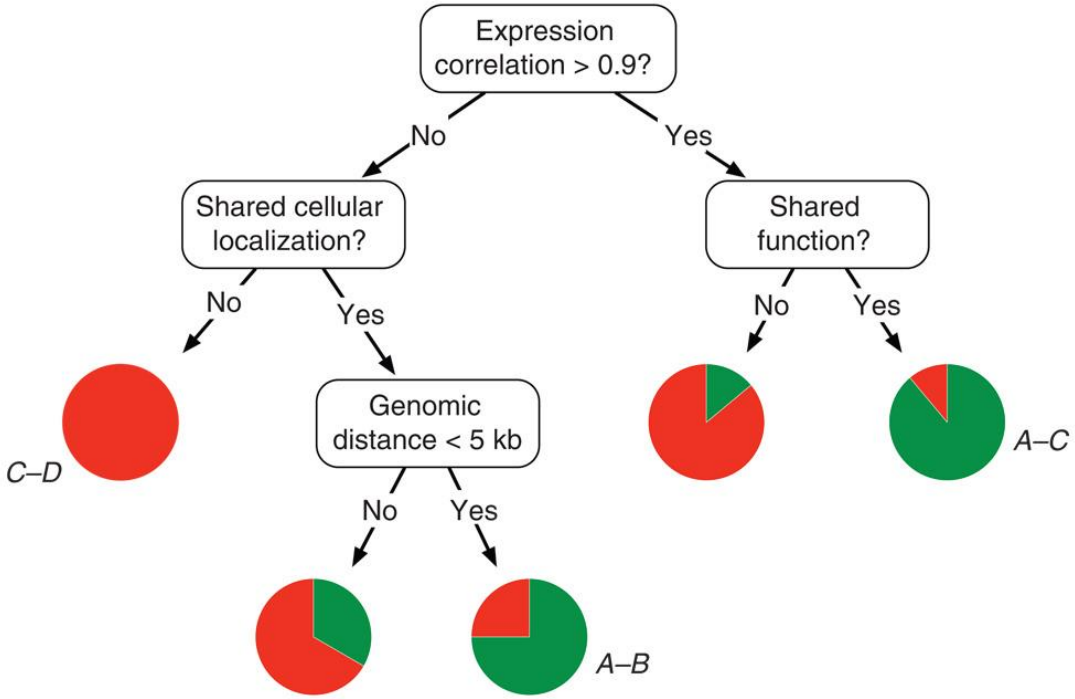
https://en.wikipedia.org/wiki/Decision_tree_learning

Decision tree example for predicting protein-protein interactions

a

Gene Pair	Interact?	Expression correlation	Shared localization?	Shared function?	Genomic distance
A-B	Yes	0.77	Yes	No	1 kb
A-C	Yes	0.91	Yes	Yes	10 kb
C-D	No	0.1	No	No	1 Mb
⋮					

b



https://pmc.ncbi.nlm.nih.gov/articles/PMC2701298/

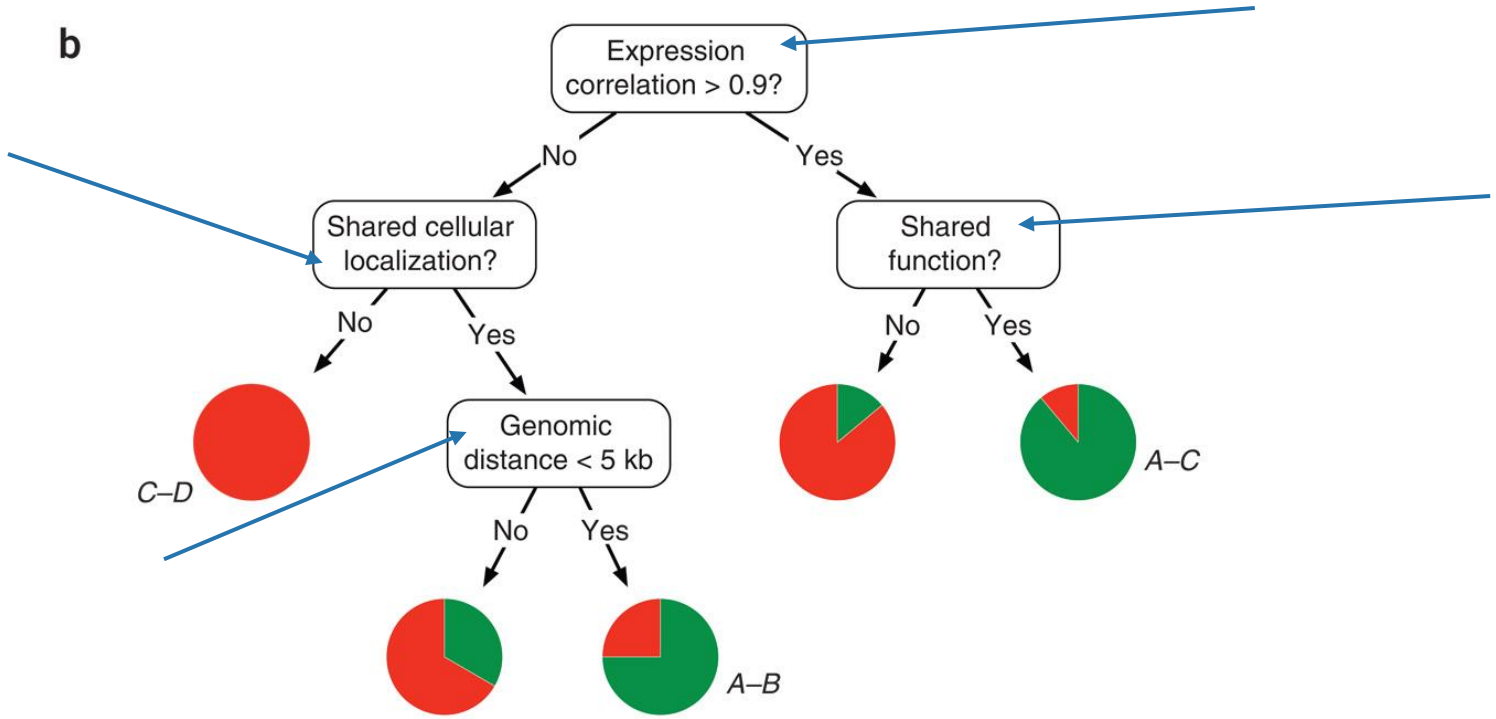
Decision tree example for predicting protein-protein interactions

a

Gene Pair	Interact?	Expression correlation	Shared localization?	Shared function?	Genomic distance
A-B	Yes	0.77	Yes	No	1 kb
A-C	Yes	0.91	Yes	Yes	10 kb
C-D	No	0.1	No	No	1 Mb
⋮					

How are these conditions determined and in what order to evaluate these conditions?

b



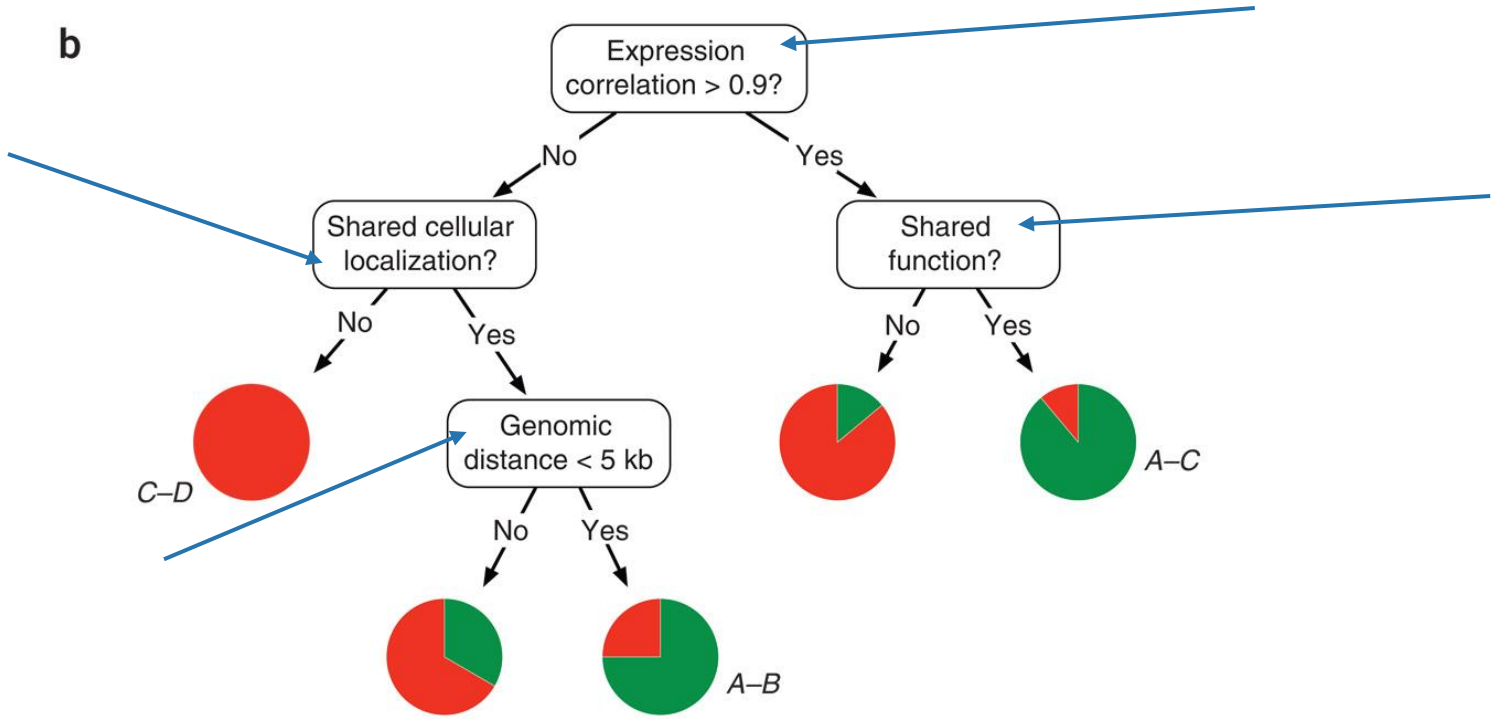
https://pmc.ncbi.nlm.nih.gov/articles/PMC2701298/

Decision tree example for predicting protein-protein interactions

a

Gene Pair	Interact?	Expression correlation	Shared localization?	Shared function?	Genomic distance
A-B	Yes	0.77	Yes	No	1 kb
A-C	Yes	0.91	Yes	Yes	10 kb
C-D	No	0.1	No	No	1 Mb
⋮					

b



How are these conditions determined and in what order to evaluate these conditions?

Common approach, though not necessary, is to use **Shannon's entropy**

https://pmc.ncbi.nlm.nih.gov/articles/PMC2701298/

Decision tree split point determination

Let S denote the entire dataset and X denote a feature considered to be split on

https://towardsdatascience.com/entropy-how-decision-trees-make-decisions-2946b9c18c8

Decision tree split point determination

Let S denote the entire dataset and X denote a feature considered to be split on

Information gain $\coloneqq IG(S, X) = H(S) - H(S|X)$

https://towardsdatascience.com/entropy-how-decision-trees-make-decisions-2946b9c18c8

Decision tree split point determination

Let S denote the entire dataset and X denote a feature considered to be split on

Information gain $:= IG(S,X) = \underbrace{H(S)}_{\text{Shannon Entropy before splitting data}} - \underbrace{H(S|X)}_{\text{Shannon Entropy conditioned on splitting data by feature } X}$

https://towardsdatascience.com/entropy-how-decision-trees-make-decisions-2946b9c18c8

Decision tree split point determination

Let S denote the entire dataset and X denote a feature considered to be split on

Information gain $\coloneqq IG(S, X) = H(S) - H(S|X)$

https://towardsdatascience.com/entropy-how-decision-trees-make-decisions-2946b9c18c8

Decision tree split point determination

Let S denote the entire dataset and X denote a feature considered to be split on

Information gain $:= IG(S, X) = H(S) - H(S|X)$

→ Split data such that is $IG(S, X)$ maximized

https://towardsdatascience.com/entropy-how-decision-trees-make-decisions-2946b9c18c8

Decision tree split point determination

Let S denote the entire dataset and X denote a feature considered to be split on

Information gain $:= IG(S, X) = H(S) - H(S|X)$

→ Split data such that is $IG(S, X)$ maximized

Note: **Gini** is often used instead of **Entropy**

https://towardsdatascience.com/entropy-how-decision-trees-make-decisions-2946b9c18c8

Decision tree tradeoffs

- Pros:
  - Simple to implement, understand, and interpret
  - Can handle both **numerical** and **categorical** data
  - Can be used for both **classification** and **regression**
    - Note: for **regression**, we typically use **MSE** (or something similar) instead of **Shannon Entropy**
  - Built in feature selection

https://en.wikipedia.org/wiki/Decision_tree_learning

Decision tree tradeoffs

- Pros:
  - Simple to implement, understand, and interpret
  - Can handle both **numerical** and **categorical** data
  - Can be used for both **classification** and **regression**
    - Note: for **regression**, we typically use **MSE** (or something similar) instead of **Shannon Entropy**
  - Built in feature selection
- Cons:
  - Constructing a tree is not guaranteed to be optimally fit due to **greedy** nature
  - Decision tree structures are extremely sensitive to small changes in training data
  - Very prone to overfitting

https://en.wikipedia.org/wiki/Decision_tree_learning

Decision tree tradeoffs

- Pros:
  - Simple to implement, understand, and interpret
  - Can handle both **numerical** and **categorical** data
  - Can be used for both **classification** and **regression**
    - Note: for **regression**, we typically use **MSE** (or something similar) instead of **Shannon Entropy**
  - Built in feature selection
- Cons:
  - Constructing a tree is not guaranteed to be optimally fit due to **greedy** nature
  - Decision tree structures are extremely sensitive to small changes in training data
  - Very prone to overfitting

→ **Random forests**

https://en.wikipedia.org/wiki/Decision_tree_learning

Random forests

Random forest (RF) definition

- An ensemble supervised learning approach that uses multiple decision trees trained on various subsets of the data obtained *via* bootstrapping

https://en.wikipedia.org/wiki/Random_forest

Random forest (RF) definition

- An ensemble supervised learning approach that uses multiple decision trees trained on various subsets of the data obtained *via* bootstrapping

- For **classification**, the final predicted value is *typically* the class selected by the most trees (majority voting)

- For **regression**, the final predicted value is *typically* the average value of what the trees predict

https://en.wikipedia.org/wiki/Random_forest

What makes random forests ‘random’?

- **Bootstrapping** training data: each tree in the forest is trained on a random subset of the training data (with replacement), and then final predictions are an **aggregation** of the trees’ predictions a process known as **bagging**

https://en.wikipedia.org/wiki/Random_forest

What makes random forests ‘random’?

- **Bootstrapping** training data: each tree in the forest is trained on a random subset of the training data (with replacement), and then final predictions are an **aggregation** of the trees’ predictions a process known as **bagging**

- Random feature selection: at each node in a tree, only a random subset of features is considered for splitting, introducing randomness in tree construction and reducing correlation between trees (**feature bagging**)

https://en.wikipedia.org/wiki/Random_forest

Random forest hyperparameters

- Number of trees: determines the size of the forest and influences model stability

https://towardsdatascience.com/hyperparameter-tuning-the-random-forest-in-python-using-scikit-learn-28d2aa77dd74

Random forest hyperparameters

- Number of trees: determines the size of the forest and influences model stability
- Number of features sampled per split: adds randomness to tree construction for better generalization

https://towardsdatascience.com/hyperparameter-tuning-the-random-forest-in-python-using-scikit-learn-28d2aa77dd74

Random forest hyperparameters

- Number of trees: determines the size of the forest and influences model stability
- Number of features sampled per split: adds randomness to tree construction for better generalization
- Tree-specific parameters: depth, minimum samples per split, and other controls for tree complexity

https://towardsdatascience.com/hyperparameter-tuning-the-random-forest-in-python-using-scikit-learn-28d2aa77dd74

Random forest hyperparameters

- Number of trees: determines the size of the forest and influences model stability
- Number of features sampled per split: adds randomness to tree construction for better generalization
- Tree-specific parameters: depth, minimum samples per split, and other controls for tree complexity
- Bootstrapping settings: how to sample and subset sizes for data and features

https://towardsdatascience.com/hyperparameter-tuning-the-random-forest-in-python-using-scikit-learn-28d2aa77dd74

Random forest hyperparameters

- Number of trees: determines the size of the forest and influences model stability
- Number of features sampled per split: adds randomness to tree construction for better generalization
- Tree-specific parameters: depth, minimum samples per split, and other controls for tree complexity
- Bootstrapping settings: how to sample and subset sizes for data and features
- And more!

https://towardsdatascience.com/hyperparameter-tuning-the-random-forest-in-python-using-scikit-learn-28d2aa77dd74

Random forest hyperparameters

- Number of trees: determines the size of the forest and influences model stability
- Number of features sampled per split: adds randomness to tree construction for better generalization
- Tree-specific parameters: depth, minimum samples per split, and other controls for tree complexity
- Bootstrapping settings: how to sample and subset sizes for data and features
- And more!

→ Typically use approaches like **cross validation** or **out-of-bag** (**OOB**) error to perform tuning

https://towardsdatascience.com/hyperparameter-tuning-the-random-forest-in-python-using-scikit-learn-28d2aa77dd74

RF implementation in R

```
# Example with Iris dataset
library(randomForest);
data(iris);

set.seed(123);

rf.model <- randomForest(
    Species ~ .,
    data = iris,
    importance = TRUE
    );

rf.model;
```


https://www.rdocumentation.org/packages/randomForest/versions/4.7-1.2/topics/randomForest

RF implementation in R

```
Call:
 randomForest(formula = Species ~ ., data = iris, importance =
TRUE)
               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 2

        OOB estimate of  error rate: 4.67%
Confusion matrix:
           setosa versicolor virginica class.error
setosa         50          0         0        0.00
versicolor      0         47         3        0.06
virginica       0          4        46        0.08
```


https://www.rdocumentation.org/packages/randomForest/versions/4.7-1.2/topics/randomForest

RF implementation in R with `ntree` and `mtry` tuning using grid search

```
tune.rf <- function(data, formula, ntree.values, mtry.values, seed = 123) {
    results <- expand.grid(ntree = ntree.values, mtry = mtry.values);
    results$oob_error <- NA;
    for (i in 1:nrow(results)) {
        set.seed(seed);
        rf.model <- randomForest(
            formula = formula,
            data = data,
            ntree = results$ntree[i],
            mtry = results$mtry[i],
            importance = TRUE
            );
        results$oob_error[i] <- rf.model$err.rate[results$ntree[i], 'OOB'];
        }
    return(results);
    }
```


https://www.rdocumentation.org/packages/randomForest/versions/4.7-1.2/topics/randomForest

RF implementation in R with `ntree` and `mtry` tuning using grid search

```
# Tuning ntree and mtry hyperparameters
ntree.values <- seq(10, 500, by = 10);
mtry.values <- 1:4;
tuning.result <- tune.rf(
    data = iris,
    formula = Species ~ .,
    ntree.values = ntree.values,
    mtry.values = mtry.values
    );

optimal.params <- tuning.result[which.min(tuning.result$oob_error), ];

# Fitting a tuned model
set.seed(123);
tuned.model <- randomForest(
    Species ~ .,
    data = iris,
    ntree = optimal.params$ntree,
    mtry = optimal.params$mtry,
    importance = TRUE
    );
tuned.model;
```


https://www.rdocumentation.org/packages/randomForest/versions/4.7-1.2/topics/randomForest

RF implementation in R with `ntree` and `mtry` tuning using grid search

```
Call:
 randomForest(formula = Species ~ ., data = iris, ntree =
optimal.params$ntree,      mtry = optimal.params$mtry, importance = TRUE)
               Type of random forest: classification
                     Number of trees: 20
No. of variables tried at each split: 2

        OOB estimate of  error rate: 3.33%
Confusion matrix:
           setosa versicolor virginica class.error
setosa         50          0         0        0.00
versicolor      0         47         3        0.06
virginica       0          2        48        0.04
```


https://www.rdocumentation.org/packages/randomForest/versions/4.7-1.2/topics/randomForest

Questions?