Random Forest from scratch

Francesco Stucci

Dataset

(https://www.kaggle.com/zynicide/wine-reviews)

Dataset head

country	points	price	variety	winery
US	88.0	32.0	Zinfandel	Dunbar
US	92.0	50.0	Pinot Noir	Fess Parker
US	87.0	44.0	Pinot Noir	Dierberg
Italy	88.0	50.0	Nebbiolo	Amalia
US	92.0	60.0	Cabernet Sauvignon	J. Lohr

Dataset shape: (104183, 5)

The goal

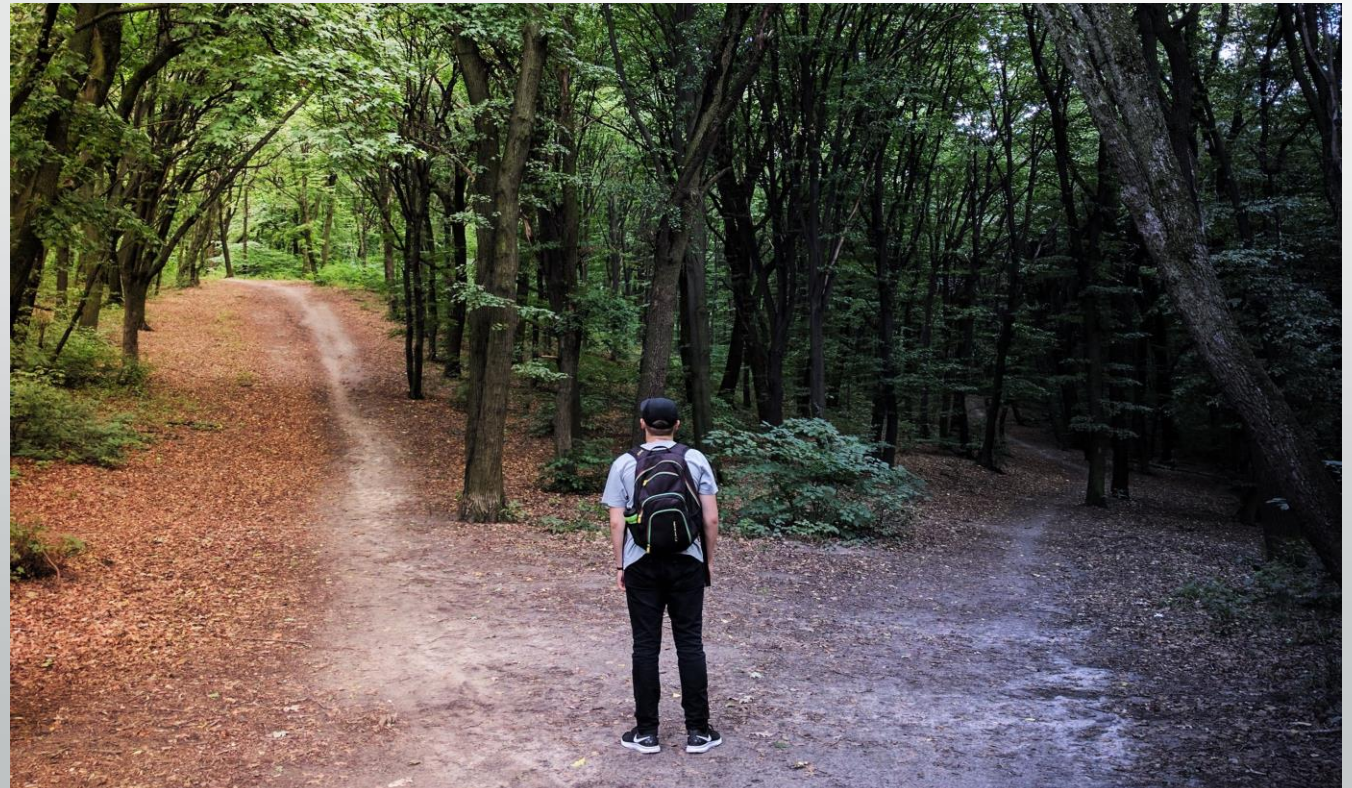
Our goal is to classify the **Variety** of wine based on the dataset features **Country, Points, Price** and **Winery**

Which model are we supposed to use?

Our data consists of labled values and we need to classify categorical values.
We have more than 100 thousend samples and low-dimensionality (just 4 dimensions).
For these reasons I chose the **Random Forest** model to deal with it

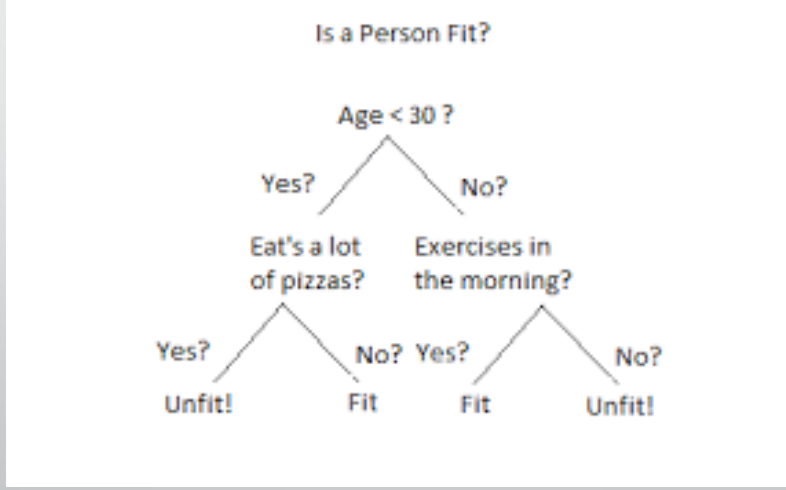
To build a Forest, we need Trees!

In order to build a **Random Forest**, we need first to define our **Decision Tree**



What a Decision Tree is

Decision Tree is a binary tree structure where each node corresponds to a question, whereas its children are the split dataset which correspond to the true or false response to the question



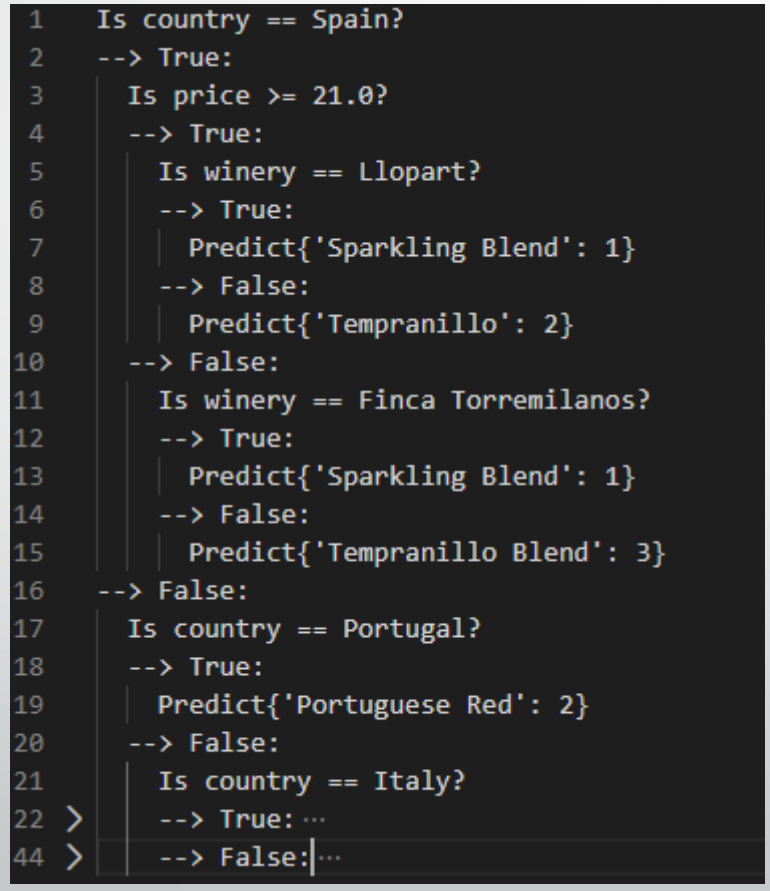
But the question is: which is the question?

We need to choose each node which question corresponds to. To select the question we have two main method

- Gini impurity: $I_G(n) = 1 - \sum_{i=1}^{J} (p_i)^2$
- Entropy: $H(X) = H(p_1, ..., p_n) = -\sum_{i=1}^{n} p_i \log_2 p_i$

By comparing these values between all question results, we choose the best one, which means the one which give us more information

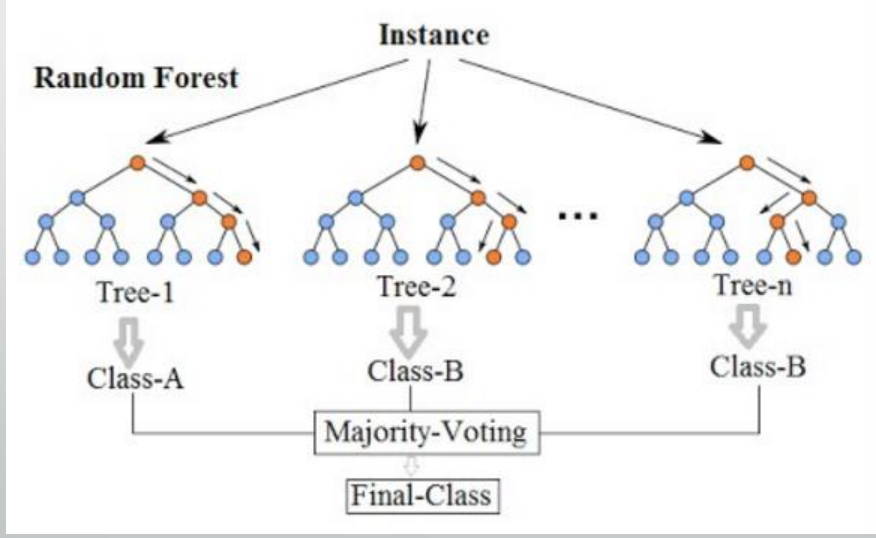
Our Decision Tree appearence



```
Is country == Spain?
--> True:
  Is price >= 21.0?
  --> True:
    Is winery == Llopart?
    --> True:
      Predict{'Sparkling Blend': 1}
    --> False:
      Predict{'Tempranillo': 2}
  --> False:
    Is winery == Finca Torremilanos?
    --> True:
      Predict{'Sparkling Blend': 1}
    --> False:
      Predict{'Tempranillo Blend': 3}
--> False:
  Is country == Portugal?
  --> True:
    Predict{'Portuguese Red': 2}
  --> False:
    Is country == Italy?
    --> True: ···
    --> False:···
```

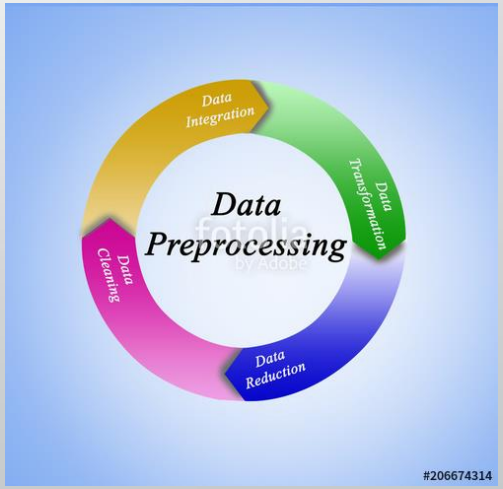
From Tree to Forest

Once we have our **Decision Tree**, to build a **Random Forest** we need to build more trees and ensemble them.
To do so, we just give each tree a random subset of the whole dataset, with a random subset of features (in order to decrese the similarity of our trees and avoid overfitting) and finally choose the most voted category from our tree predictions



Data preprocessing

Machine learning models, usually, need some data analysis in order to find out missing data, outlayers and transform categorical values in numbers.
However **Random Forest** is able to deal with these problems natively.
In particular, on this dataset, the only preprocessing perfomed was to remove samples with missing price due to its fundamental impact.



Parameters

We have to tune few parameters in order to prevent overfitting and improve accuracy of our model

- Number of trees
- Number of features for each tree
- Number of semples in each tree

Some results

With
#samples = #dataset_samples / log2(#trees)
#features = #dataset_ features

- Decision Tree Accuracy: 0.1935
- Random Forest (10 trees): 0.2215
- Random Forest (30 trees): 0.265
- Random Forest (60 trees): 0.273
- Random Forest (100 trees): 0.28

With
#samples = #dataset_samples / log2(#trees)
#features = log2(#dataset_ features)

- Decision Tree Accuracy: 0.1249
- Random Forest (10 trees): 0.266
- Random Forest (30 trees): 0.275
- Random Forest (60 trees): 0.267
- Random Forest (100 trees): 0.273

Some results

With
#samples = #dataset_samples / #trees
#features = #dataset_ features

- Decision Tree Accuracy: 0.2665
- Random Forest (10 trees): 0.278
- Random Forest (30 trees): 0.2515
- Random Forest (60 trees): 0.251
- Random Forest (100 trees): 0.2285

With
#samples = #dataset_samples / #trees
#features = log2(#dataset_ features)

- Decision Tree Accuracy: 0.257
- Random Forest (10 trees): 0.124
- Random Forest (30 trees): 0.186
- Random Forest (60 trees): 0.1525
- Random Forest (100 trees): 0.148

Comparing with SKLearn

The Random Forest implementation on this library is awfully faster as well as more accurate because of its optimizations...
(although my implementation can deal with categorical values 😆)

- **Decision Tree Accuracy:** 0.256
- **Random Forest (10 trees):** 0.297
- **Random Forest (30 trees):** 0.301
- **Random Forest (60 trees):** 0.301
- **Random Forest (100 trees):** 0.301