Python for Data Science

Machine Learning 2

ENSAE-ENSAI
Formation continue
(Cepe)

Decision Trees

Ensemble Trees

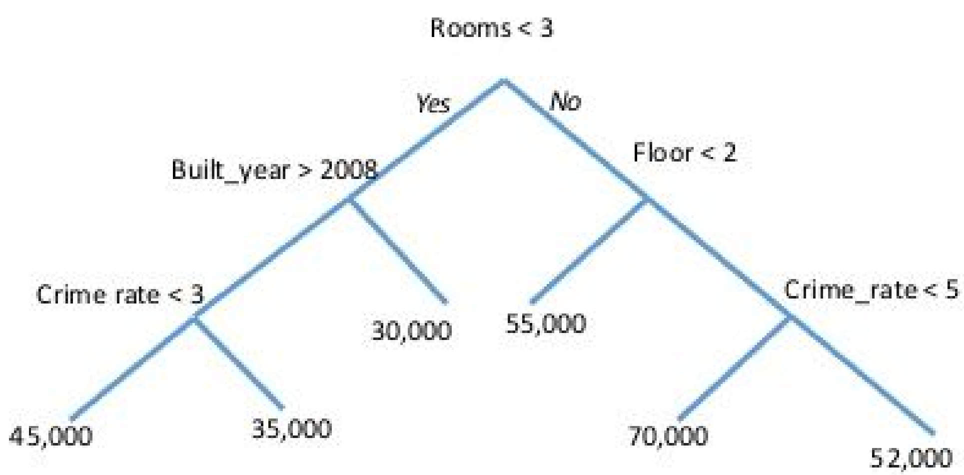
Classification

Wrap Up

Decision Trees

- Predicting Model that is based on a tree

- "Simplified algo":
  - Select most correlated variable (to target)
  - Select where to split (which value) in order to maximize **separation**
  - Decide when to stop



http://scikit-learn.org/stable/modules/tree.html

Decision Trees

We can split until every observation is separated in the tree

- But should we?

- What will be the RSS?
  - For the training set?
  - For the testing set?

Decision Trees

We can split until every observation is separated in the tree

➢ But should we?

➢ What will be the RSS?

For the training set?

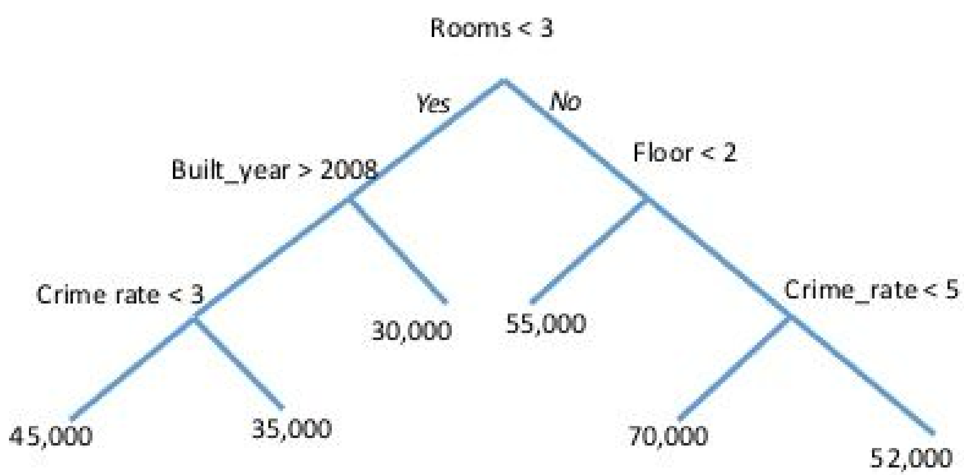
For the testing set?

➡ **Overfitting!**

Decision Trees

- How to use it?

  - Decision tree is in fact a list of conditions (if … Then … Else …)
  - Follow the steps of the tree to get a prediction

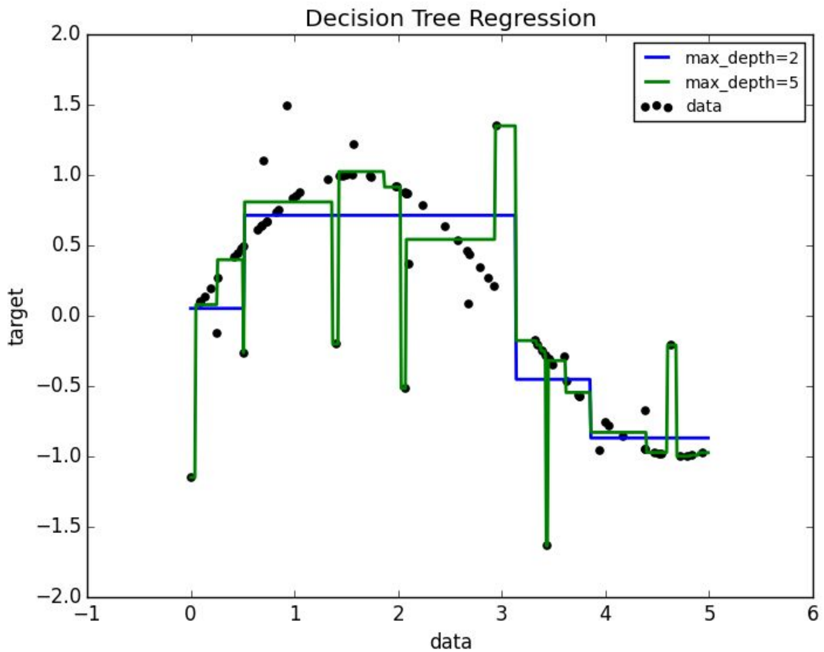


http://scikit-learn.org/stable/modules/tree.html

Decision Trees

- Pros:
  - Easy to understand and interpret
  - Testing is easy and quick

- Cons:
  - Bad at generalising (overfitting)
  - Unstable
  - Training is not abvious:
    - Approaches: ID3, C4.5, C5.0 and CA



Decision Tree Regression

```
from sklearn.tree import DecisionTreeRegressor
X = [[0, 0], [2, 2]]
y = [0.5, 2.5]
clf = DecisionTreeRegressor(
    max_depth=10,
    # minimum number of samples required for a split
    min_samples_split=5
    )
clf.fit(X, y)
clf.predict([[1, 1]])
```

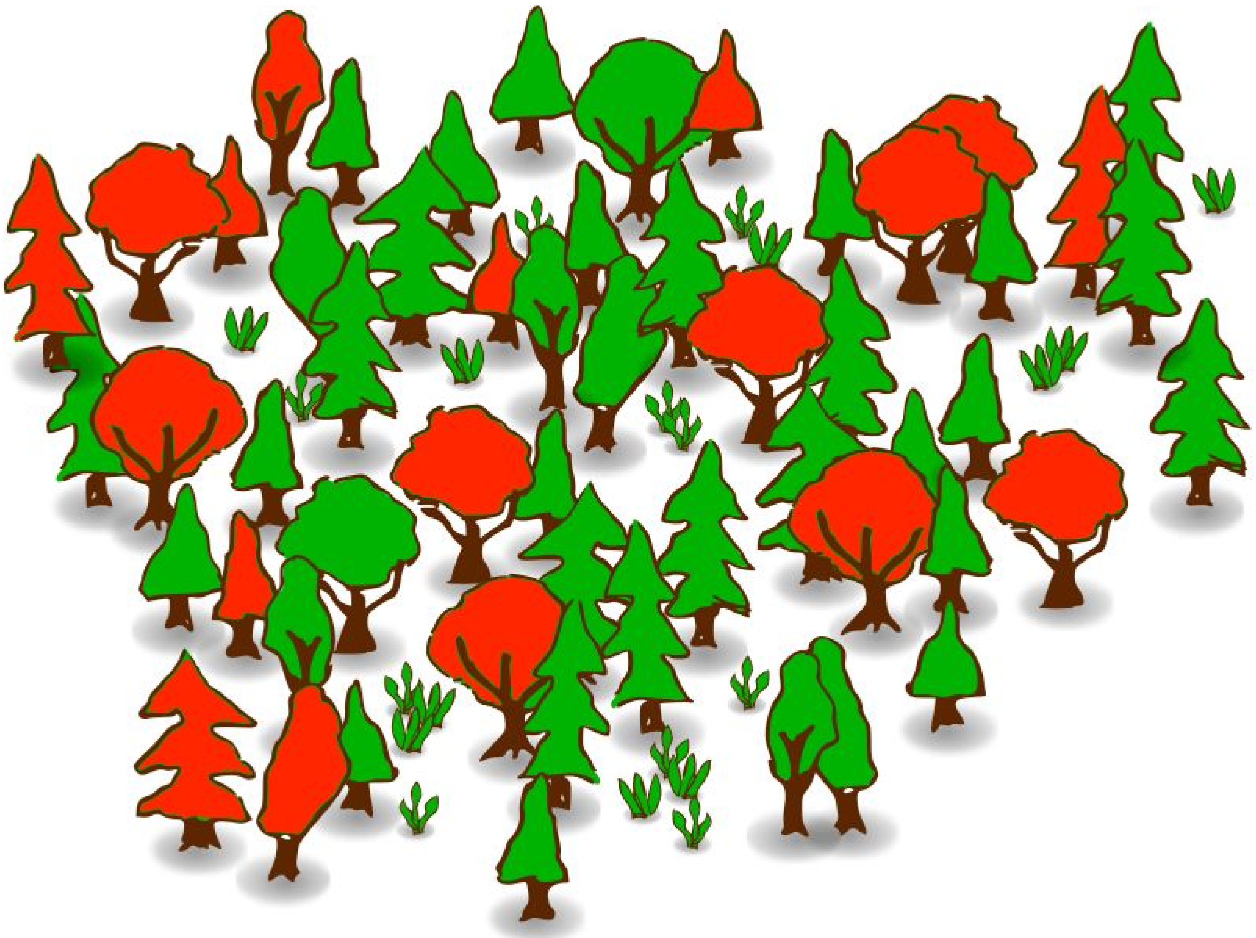
Decision Trees

Ensemble Trees

Classification

Wrap Up

Ensemble Trees



Ensembling Techniques

Bagging:

- **Sample** the input data (features) to generate multiple sets of input data.
  - Usually done with replacement
  - Size of samples is similar to the original data
    - Useful to reduce variance without increasing the bias

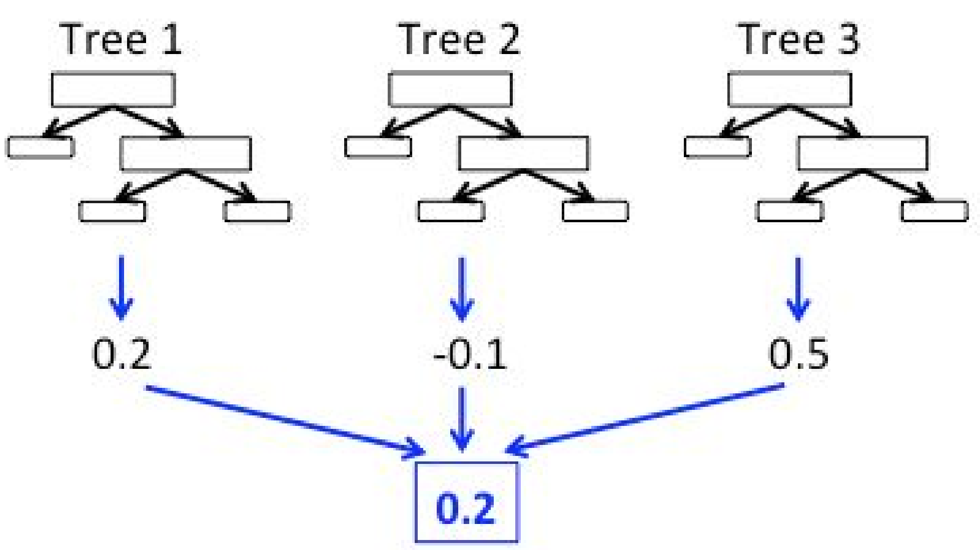
Frodo Baggins

Boosting

- Mix a set of **weak** learners to build a strong learner
  - Learners are simple (example decision tree stump)
  - Each weak learner has low variance but high bias
    - Useful to reduce bias without increasing variance

Random Forest

- Bagging is similar to an "artificial" increase in the training set:
  - We create n decision trees, We train each decision tree separately:
    - Training data: draw a sample subset from the training data. Sub-sample size is same as the original input sample size but samples are drawn with replacement
    - Select randomly ***d*** features (usually d=√m) without replacement
    - Splits are minimizing gini or entropy
  - Calculate the mean to give the final prediction



Random Forests in Scikit-Learn

```python
from sklearn.ensemble import RandomForestClassifier

model = RandomForestClassifier(n_estimators=10)

# Train the model using the training set
model.fit(X_train, y_train)

# Predict target for the testing set
y_hat = model.predict(X_test)
```

Random Forests

- Pros:
  - Don’t overfit
  - Easy to tune
  - Trees grow in parallel - Fast!
  - No overfitting Danger
  - Ideal to estimate quickly predictability

- Cons:
  - “Black box” models, Not easy to interpret
  - Can be complex to deploy (over SQL, Excel, …)

Quick Solution (Good most of the time!)

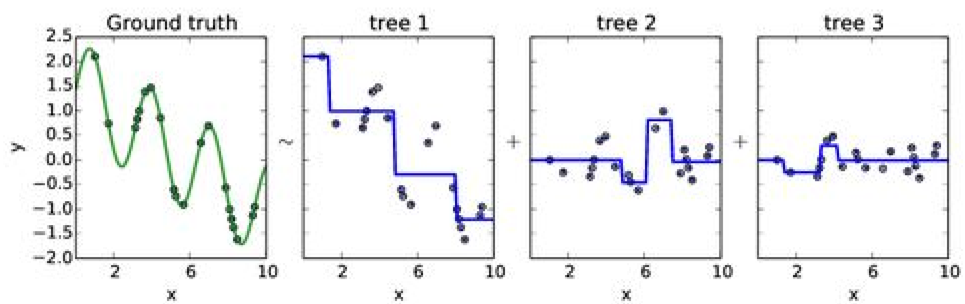
Gradient Boosting

- The algorithm constructs the trees sequentially (slow learning)
- Each tree is grown using the information from previous tree
- Typically the tree depth used is smaller than for Random Forests

Gradient Boosting

- The algorithm constructs the trees sequentially (slow learning)
- Each tree is grown using the information from previous tree
- Typically the tree depth used is smaller than for Random Forests

1. Set $\hat{f}(x) = 0$ and $r_i = y_i$ for all i in the training set.
2. For $b = 1, 2, \ldots, B$, repeat:
   2.1 Fit a tree $\hat{f}^b$ with d splits ($d+1$ terminal nodes) to the training data (X, r).
   2.2 Update $\hat{f}$ by adding in a shrunken version of the new tree:
   $$\hat{f}(x) \leftarrow \hat{f}(x) + \lambda \hat{f}^b(x).$$
   2.3 Update the residuals,
   $$r_i \leftarrow r_i - \lambda \hat{f}^b(x_i).$$
3. Output the boosted model,
$$\hat{f}(x) = \sum_{b=1}^{B} \lambda \hat{f}^b(x).$$



Gradient Boosting

- Pros:
  - Highest performance

- Cons:
  - Can’t be parallelized (Solution: Parallel implementation such as XGBoost)
  - Many parameters to tune (more “Data Scientist time” is necessary)
  - Overfitting Danger!
  - Same disadvantages as Random Forest

↓

Highest Performance (Requires more Work!)

Gradient Boosting in Scikit-Learn

```python
from sklearn.ensemble import GradientBoostingClassifier

model = GradientBoostingClassifier(n_estimators=100)

# Train the model using the training set
model.fit(X_train, y_train)

# Predict target for the testing set
y_hat = model.predict(X_test)
```

Decision Trees

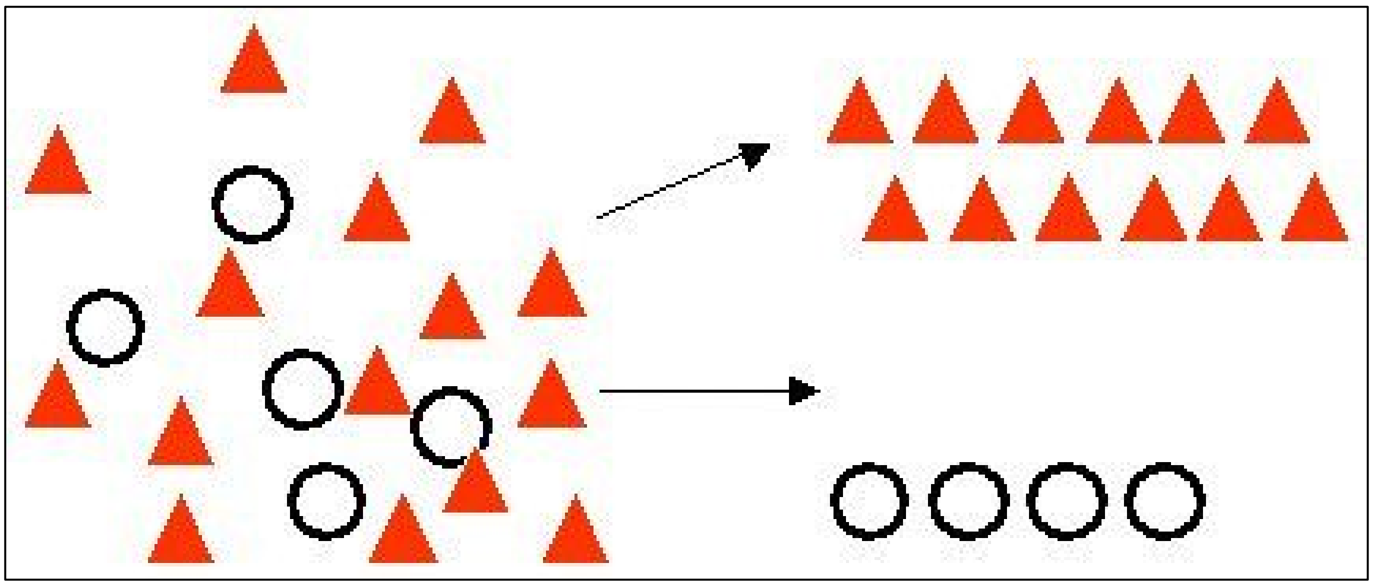
Ensemble Trees

Classification

Wrap Up

Classification

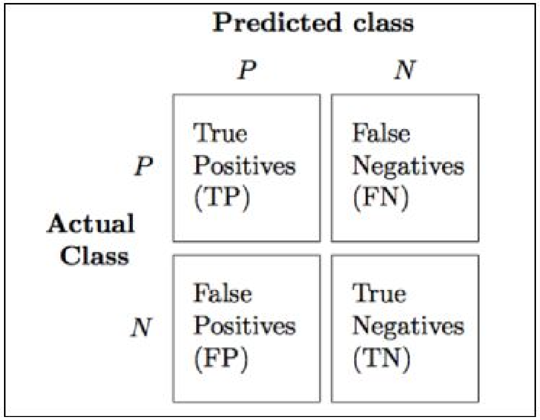
❖ Which category a new observation belongs to?



Evaluation

- Confusion Matrix - Classification **performance** report

```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_true, y_pred)
```



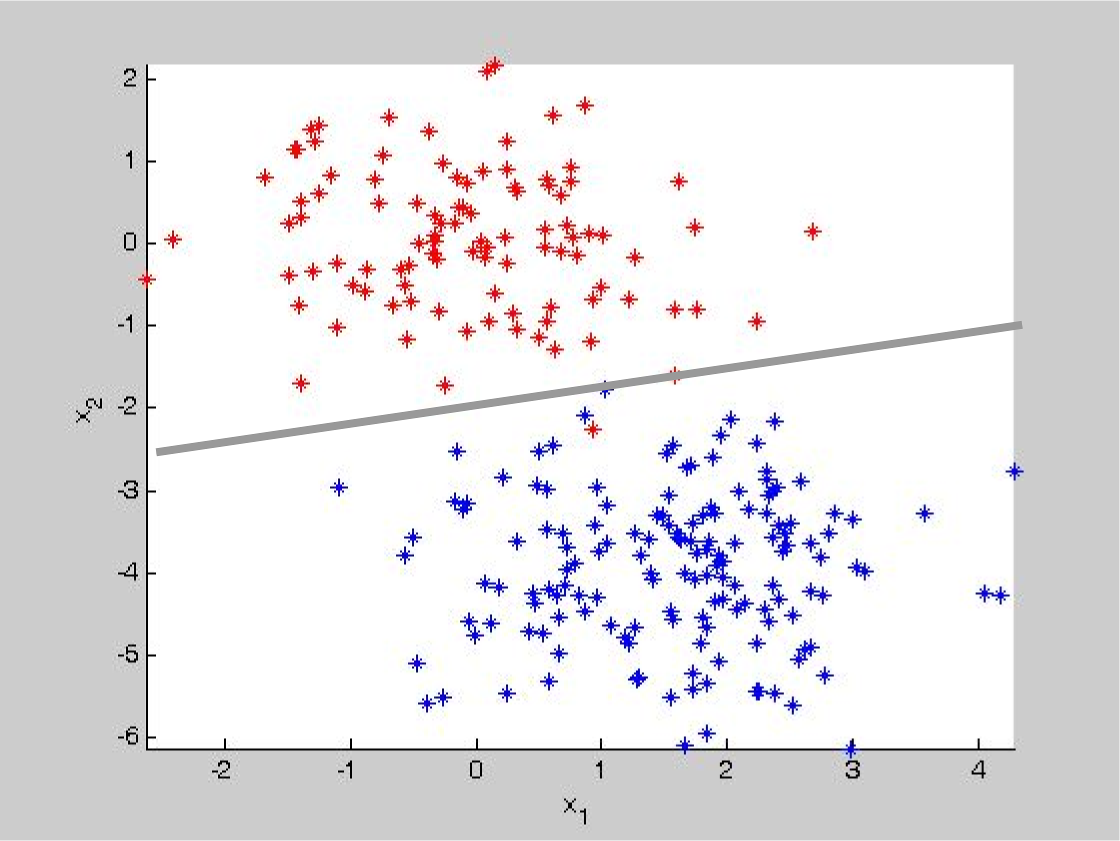
- Accuracy - How often the prediction is correct?

Accuracy = (TP + TN)/ Total

```
from sklearn.metrics import accuracy_score
accuracy_score(y_true, y_pred)
```

Logistic Regression

❖ Finding the linear curve that seperates the observations into classes:

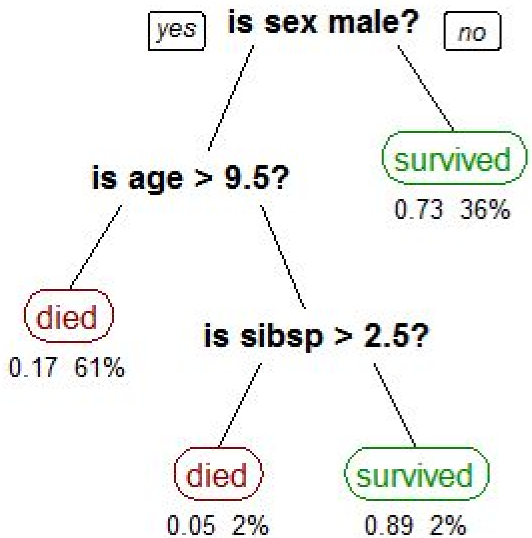


Logistic Regression in Scikit-Learn

```
from sklearn.linear_model import LogisticRegression

logreg = LogisticRegression(penalty='l2', C=1.0)

# Train the model using the training set
logreg.fit(X_train, y_train)


# Predict target for the testing set
y_hat = logreg.predict(X_test)
```

Decision Tree Classifier

❖ Finding the decision tree that seperates the observations into classes:



Decision Tree Classifier in Scikit-Learn

```python
from sklearn.tree import DecisionTreeClassifier

model = DecisionTreeClassifier(max_depth=10)

# Train the model using the training set
model.fit(X_train, y_train)
```

Random Forest Classifier in Scikit-Learn

```python
from sklearn.ensemble import RandomForestClassifier

model = RandomForestClassifier(
            n_estimators=100,
            max_depth=2
            )

# Train the model using the training set
model.fit(X_train, y_train)
```

Gradient Boosting Classifier in Scikit-Learn

```python
from sklearn.ensemble import GradientBoostingClassifier

model = GradientBoostingClassifier(
           n_estimators=100,
           learning_rate=1.0,
           max_depth=2
           )

# Train the model using the training set
model.fit(X_train, y_train)
```

Decision Tree

Ensemble Trees

Classification

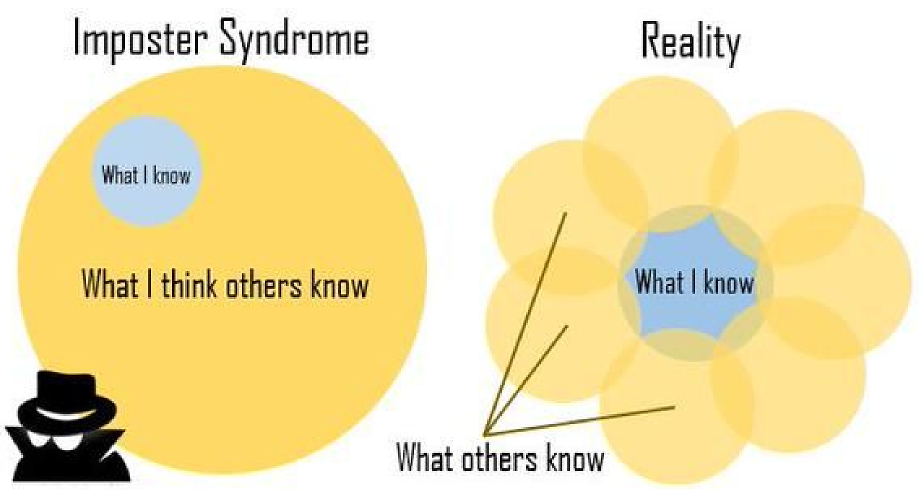
Wrap Up

Wrap-up

- I hope you think that Python and Machine Learning is fun :)
- To improve your ML skills, you should mix **theory** (MOOCs & Books) with **Kaggle** competitions / projects at **work**.
- ML is just a **part** of a Data Scientist's job - sometimes short and long-awaited ...
- You will rarely need to **implement** algorithms (unless you do research) - you must know how to **use** algorithms and **understand** how they work.
- Remember: Feature engineering is key

A few more tips...

- Join a **meetup** or create one
- Have a **TODO** list for technical books, moocs and articles
- Use **Twitter** for work
- Use **wasted time** (e.g. in public transportation) to learn stuff
- Don't forget that Data Scientists suffer from **imposter syndrome** too...



Thanks a lot!

kkarp@equancy.com

ENSAE-ENSAI
Formation continue
(Cepe)