COMP47350: Data Analytics (Conv)

Dr. Georgiana Ifrim
georgiana.ifrim@ucd.ie

Insight Centre for Data Analytics
School of Computer Science
University College Dublin

2018/19

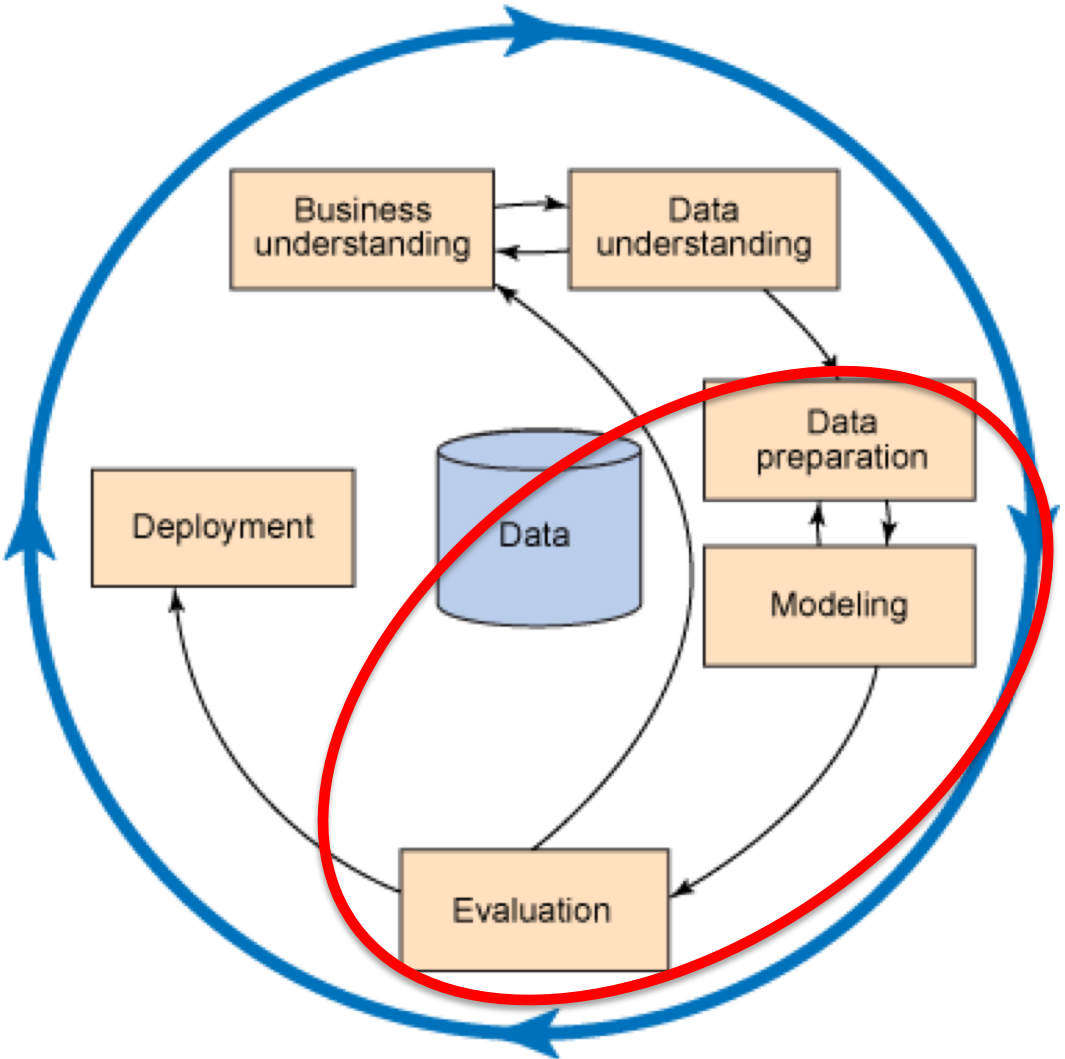
Module Topics

- **Python Environment** (Anaconda, Jupyter Notebook)
- **Getting Data** (Web scrapping, APIs, DBs)
- **Understanding Data** (slicing, visualisation)
- **Preparing Data** (cleaning, transformation)
- **Modeling & Evaluation** (machine learning)

Data Analytics Project Lifecycle:

CRISP-DM

CRISP-DM: **CR**oss-**I**ndustry **S**tandard **P**rocess for **D**ata **M**ining



Modeling Data

- Modeling:
  - How to build prediction models
  - How to evaluate prediction models

We study three supervised machine learning methods:

- Linear Regression (linear model)
- Logistic Regression (linear model)
- **Random Forests** (non-linear model)

Random Forests

- **Random Forest** = An ensemble (a committee) of Decision Tree models
- Each Decision Tree algorithm is trained on a subset of the data
- The final decision is taken by aggregating the predictions of all trained Decision Tree models (e.g., majority voting or average over probability(class=1))

Ensembling and Random Forests

- **Ensemble:** a committee of different models that are aggregated to get a final prediction

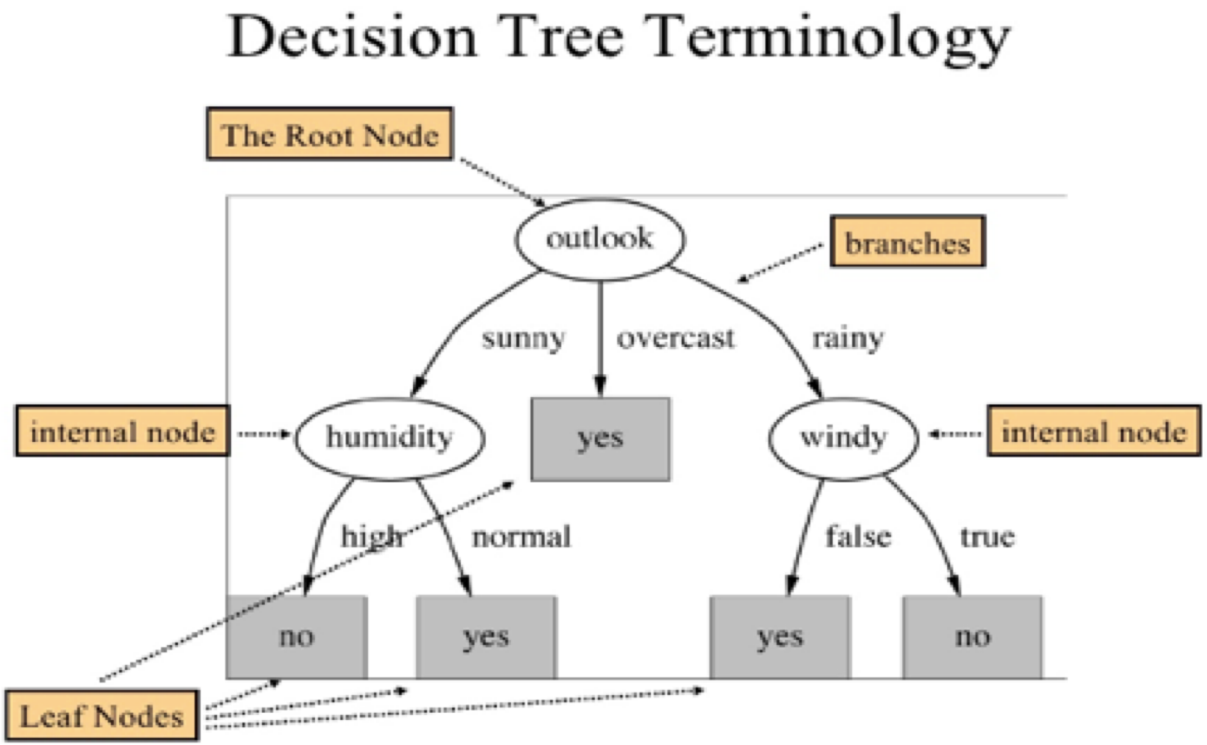
- Example of other ensembles for classification:
  - Ensemble of 3 different logistic regression models
  - Each model predicts a class for a test example (model1 predicts 1, model 2 predicts 1, model 3 predicts 0)
  - Final predicted class is established by majority voting: final predicted class is 1.

- A **Decision Tree** algorithm can be used for regression or classification
- A Decision Tree model is a set of if-then-else rules computed from the training data and used to classify the test data

- A **Random Forest (RF)** model is an ensemble of Decision Tree models
- Random Forest can be used for either classification or regression

Decision Tree

- Example decision tree for predicting class **PlayTennis** (YES, NO) based on features about the weather
- Features:
  - **Outlook** (sunny, overcast, rainy)
  - **Humidity** (high, normal)
  - **Windy** (false, true)



Classification: Example

	ID	Size	Floor	BroadbandRate	EnergyRating	PriceClass
0	1	500	4	8	C	0.0
1	2	550	7	50	A	0.0
2	3	620	9	7	A	0.0
3	4	630	5	24	B	0.0
4	5	665	8	100	C	0.0
5	6	700	4	8	B	0.0
6	7	770	10	7	B	1.0
7	8	880	12	50	A	1.0
8	9	920	14	8	C	1.0
9	10	1000	9	24	B	1.0

Can we predict the **Price Class** **(high** or **low)**, given the descriptive features (**Size, Floor, BroadbandRate, EnergyRating**) for an office?

Classification: Example

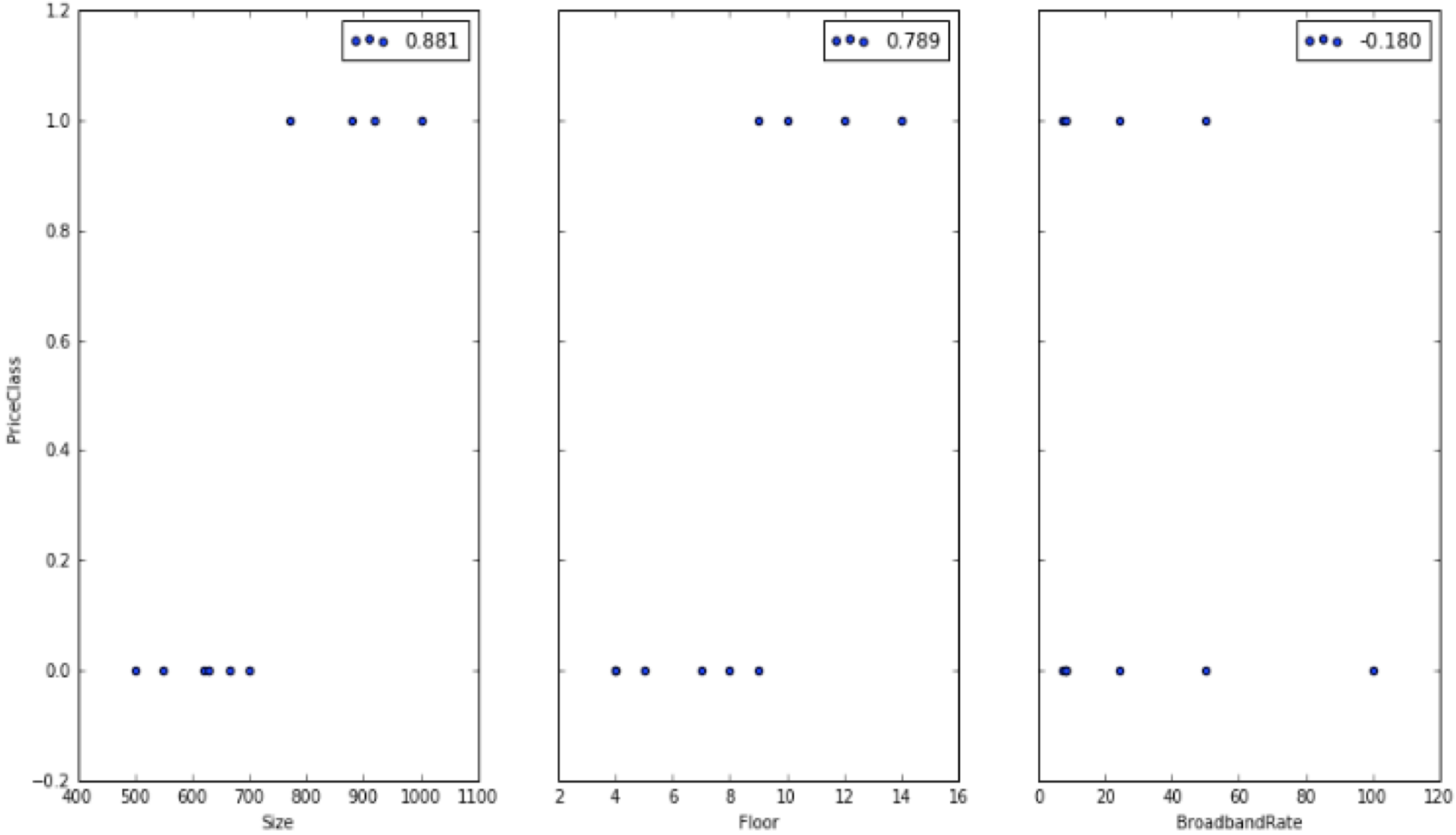
	ID	Size	Floor	BroadbandRate	EnergyRating	PriceClass
0	1	500	4	8	C	0.0
1	2	550	7	50	A	0.0
2	3	620	9	7	A	0.0
3	4	630	5	24	B	0.0
4	5	665	8	100	C	0.0
5	6	700	4	8	B	0.0
6	7	770	10	7	B	1.0
7	8	880	12	50	A	1.0
8	9	920	14	8	C	1.0
9	10	1000	9	24	B	1.0

- Which feature should we choose first to split the examples into groups with High and Low classes? How do we decide a threshold on the feature value (for continuous features)?

Understanding Data

```
# Scatterplots for each descriptive feature and target feature.
# Show the correlation value in the plot.
fig, axs = plt.subplots(1, 3, sharey=True)
df_classif.plot(kind='scatter', x='Size', y='PriceClass', label="%.3f" % df_classif[['Size', 'PriceClass']].corr()
df_classif.plot(kind='scatter', x='Floor', y='PriceClass', label="%.3f" % df_classif[['Floor', 'PriceClass']].corr
df_classif.plot(kind='scatter', x='BroadbandRate', y='PriceClass', label="%.3f" % df_classif[['BroadbandRate', 'Pr
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x110dd6be0>
```



Decision Tree: Visual Example

ID	Size	Floor	BroadbandRate	EnergyRating	PriceClass
1	500	4	8	C	0.0
2	550	7	50	A	0.0
3	620	9	7	A	0.0
4	630	5	24	B	0.0
5	665	8	100	C	0.0
6	700	4	8	B	0.0
7	770	10	7	B	1.0
8	880	12	50	A	1.0
9	920	14	8	C	1.0
10	1000	9	24	B	1.0

Start: 10 examples (6 in class 0, 4 in class 1). **Prob(class=1)** = 4/10 = 0.4

Size <= 665

Training stage: build tree model (a set of rules)

Prediction stage: apply rules on new example, and follow the splits of the tree to reach an leaf node and classify the example

True

False

5 examples (5, 0)
Prob(class=1) = 0/5 = 0
Predict class 0

5 examples (1, 4)
Prob(class=1) = 4/5 = 0.8
Floor <= 4

True

False

1 examples (1, 0)
Prob(class=1) = 0/1=0
Predict class 0

4 examples (0, 4)
Prob(class=1) = 4/4=1
Predict class 1

This model encodes if-then prediction rules, e.g.: **If Size <= 665 and Floor <= 4 then Class 0**

Decision Tree: Overview

- Key ideas to build a decision tree model:
  - Training stage finds **a set of if-then-else decision rules** based on splitting by feature values
  1. **At each level of the tree, select a single best feature** to split the examples into 2 groups, based on a fixed feature value (e.g., Size <=**665**)
     - Multiple goodness criteria for selecting a feature: reduction in error, reduction in entropy, Gini Index (selection in sklearn is based on the Gini Index)
     - The key idea is that the split leads to pure nodes (where pure means a majority of examples in that node belong to one class)
  2. **Keep selecting features until stopping criterion**, e.g.:
     - No more splits are required (groups are pure, i.e., single class in each node)
     - Reached the allowed number of splits (aka tree depth)
     - Reached the min number of examples in each group (aka leaf node size)

Decision Tree: Overview

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

- **Advantages:**
  - Highly interpretable (set of human readable if-then-else rules)
  - Can be displayed graphically
  - Prediction is fast
  - Able to directly handle multi-class problems
  - Able to handle continuous and categorical features
  - Able to capture feature interactions (non-linear model)

- **Disadvantages** (this is improved by ensembling):
  - Predictive accuracy is not as high as for some other supervised learning algorithms
  - Can easily overfit the training data (can grow big tree to perfectly fit training data)
  - Decision trees can be unstable because small variations in the data may result in a completely different tree being generated

- There are several algorithms for implementing decision trees (e.g., ID3, C4.5, CART)
- **Sklearn implementation (see link above):** "**CART** constructs binary trees using the feature and threshold that yield the largest information gain at each node. scikit-learn uses an optimised version of the CART algorithm. ...this module does not support missing values."

Random Forest

Key Idea: use multiple subsamples of the original dataset

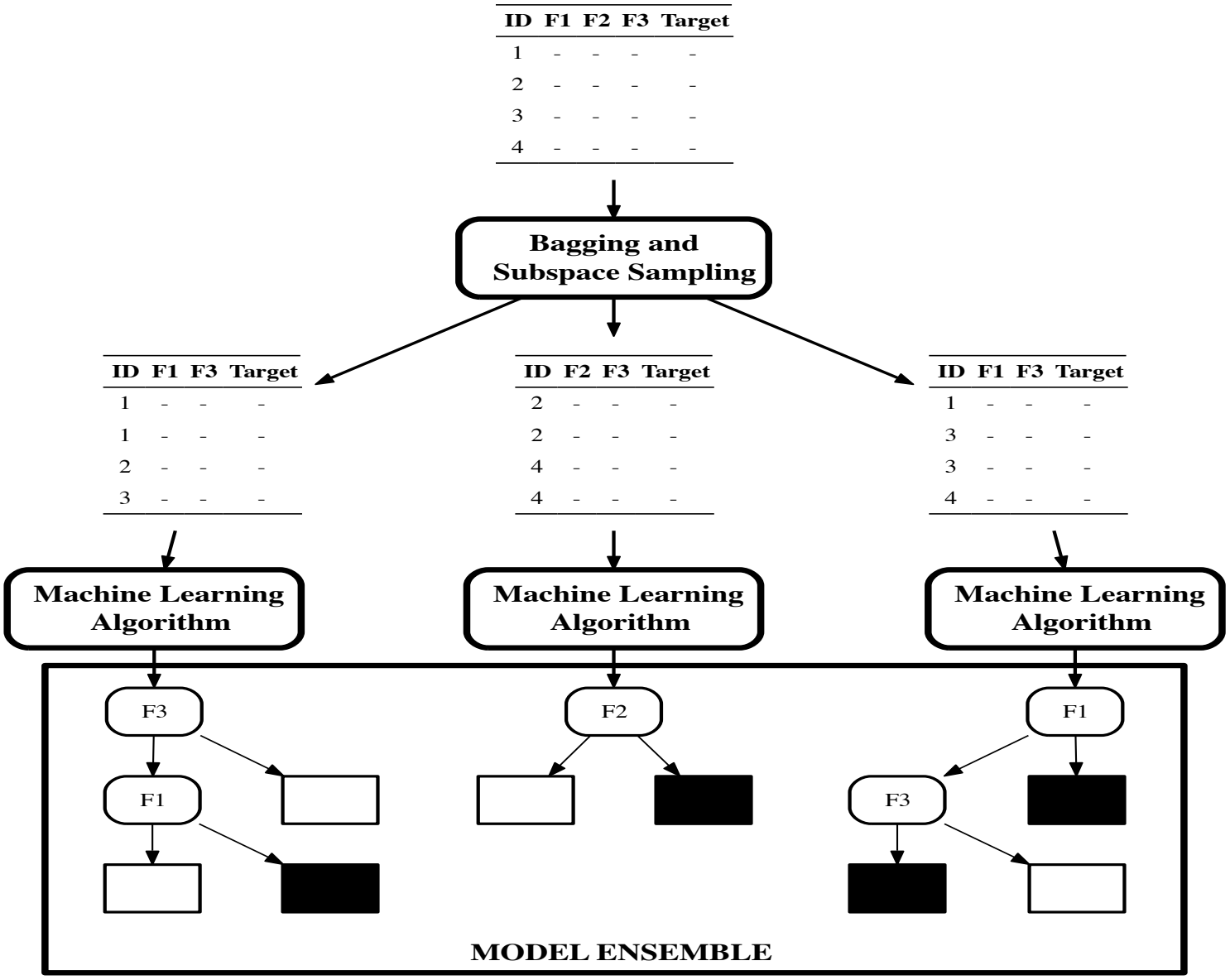
- **select random subsets of rows** (i.e., examples; aka bagging) and **select random subsets of columns** (i.e., features; aka subspace sampling); train a model for each new dataset; ensemble the trained models to give a prediction

- Example: Random Forest with 10 trees
- Create 10 new subsets of data of equal size to the original set: **select random examples**/rows with replacement for each subset (aka 10 bootstrap samples) and random subset of features

- **Training:** For each subset, fit a decision tree (results in 10 different tree models)

- **Prediction:** Ensemble predictions from the 10 decision trees, to get a prediction by majority voting

Random Forest: Visual Example



Random Forest: Overview

http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

- **Advantages:**
  - Carries over some of the advantages of the Decision Tree model regarding feature types (numeric and categorical), multi-class, etc
  - Very accurate due to the committee-voting/ensemble approach and because of using a DT as base model
  - Can be implemented in parallel for efficient training (train each base DT model independently)

- **Disadvantages**:
  - Not easy to interpret the model and the classification decision
  - Can be slow to train (time and memory requirements)
  - The trained model can be quite big, so poses challenges for deployment

- **Sklearn implementation of RF (see link above)**

Decision Tree and Random Forests: Hands-on

http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

- Training/prediction with a DT model
- Training an RF model
- Feature importance with DT and RF model
- Predicting with an RF model
- Evaluating an RF model: train/test vs cross-validation vs out-of-sample error

References

- Chapter8 from **FMLPDA Book: Fundamentals of Machine Learning for Predictive Data Analytics**, by J. Kelleher, B. Mac Namee and A. D'Arcy, MIT Press, 2015 (machinelearningbook.com)

- https://github.com/justmarkham/DAT4/blob/master/notebooks/16_ensembling.ipynb
- https://github.com/justmarkham/DAT4/blob/master/notebooks/15_decision_trees.ipynb