# Introduction to Gradient Boosting Machines

Wolfmann, Ariel
Nov, ‘17

PyData

# About me

- Computer Scientist from FaMAF, UNC. Cordoba, Argentina
- Machine Learning Developer at Machinalis
- Love to bridge the gap between Science and Business.
- LinkedIn: https://www.linkedin.com/in/ariel-wolfmann
- Twitter: @osowolfmann14
- Mail: awolfmann@machinalis.com

# Agenda

PyData San Luis 2017

- Supervised Learning
- Decision Trees
- Ensemble Methods
- Bagging
  - Random Forests
- Boosting
  - **Gradient Boosting Trees**
- Stacking
- Real life use case

# Supervised Learning

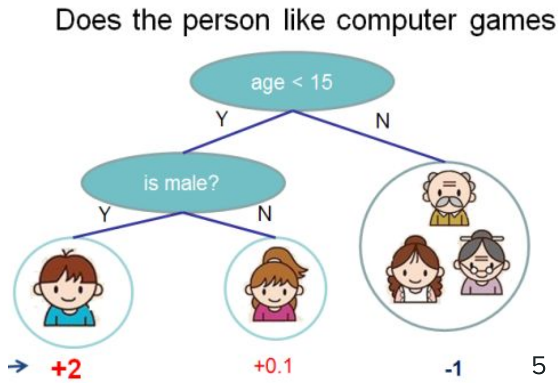
- Train a model with labelled data.
- Each data point is a pair (**x**i, yi).
  - **X**i is a feature vector -> attributes that represent an object.
  - yi is the label of the object.
- Learn to predict the label of a new data point based on what it learned from the train set.

**Classification:** predict a **discrete** variable or category.

**Regression:** predict the value of a **continuous** variable.

# Decision Tree (CART)

- **Decision Rules:** sequence of binary selections
- Can be applied to both **classification** and **regression** problems.
- Rules based on variables' values are selected to get the **best split** to differentiate observations based on the dependent variable.
- Once a rule is selected and splits a node into two, the same process is applied to each "child" node (i.e. it is a **recursive procedure**).



# Ensemble methods

PyData San Luis 2017

Combine models to improve performance.

Mix weak learners to get a strong one.

- **Bagging**
- **Boosting**
- **Stacking**

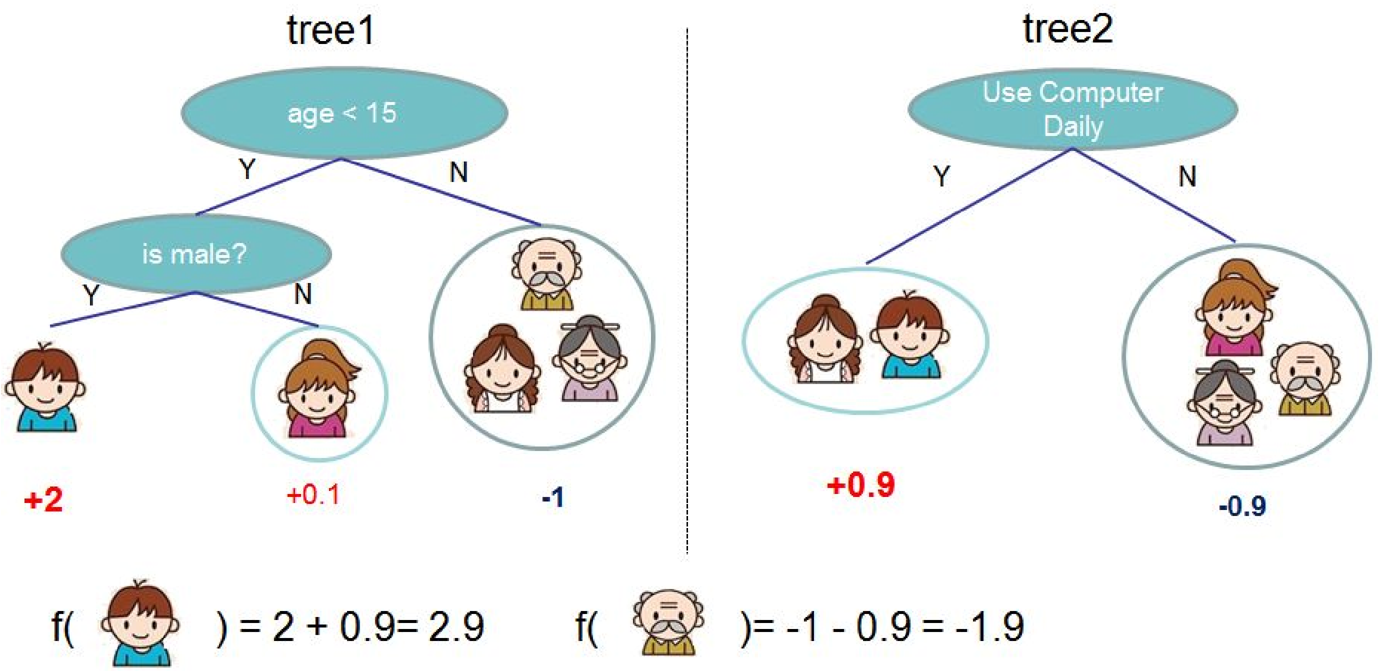
# Ensemble methods -> Bagging

- Train each weak learner in a **parallel** fashion.
- Involves having each model in the ensemble vote with **equal weight** as a “committee” and calculate the average of the predictions.
- Trains each model in the ensemble using a **randomly drawn subset of the training set.**

# Bagging -> Random Forests

- Bagging of decision trees
- **Random sample with replacement**.
- **Random subset of the features.**
- This bootstrapping procedure leads to **better model performance** because it decreases the variance of the model, without increasing the bias.

# Random Forests



# Ensemble methods -> Boosting

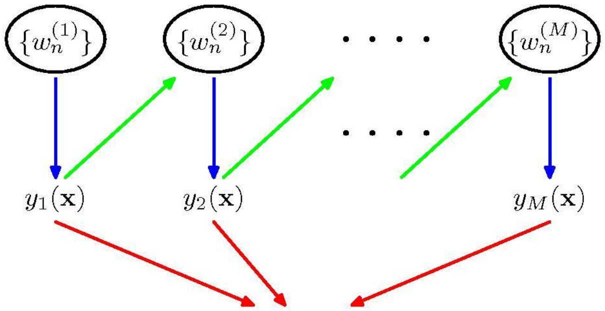
PyData San Luis 2017

- The base classifiers are trained in **sequence.**
- Each base classifier is trained using a weighted with a coefficient associated with each data point depending on the **performance of the previous classifiers**.
- **Misclassified points** by one of the base classifiers are given **greater weight** when used to train the next classifier in the sequence.

# Ensemble methods -> Boosting



- Once all the classifiers have been trained, their predictions are then combined through a **weighted majority voting scheme**.
- The subset creation is not random and depends upon the performance of the previous models: **every new subsets** contains the elements that were **misclassified by previous models.**




$$Y_M(\mathbf{x}) = \operatorname{sign}\left(\sum_{m}^{M} \alpha_m y_m(\mathbf{x})\right)$$

# Boosting -> Gradient Boosting Trees

- Boosting with CARTs as base model.
  - Add a new tree in each iteration.
  - Each tree uses information from the previous iterations.
- Uses the gradient of the loss function to minimize it.

# Implementation -> XGBoost

PyData San Luis 2017

- Open Source
- Multiple Languages: Python, R, Julia, Scala
- Performance: Multiple CPU and GPU support
- Used in most of winner solutions for ML competitions.
- Scikit-learn interface and interaction.

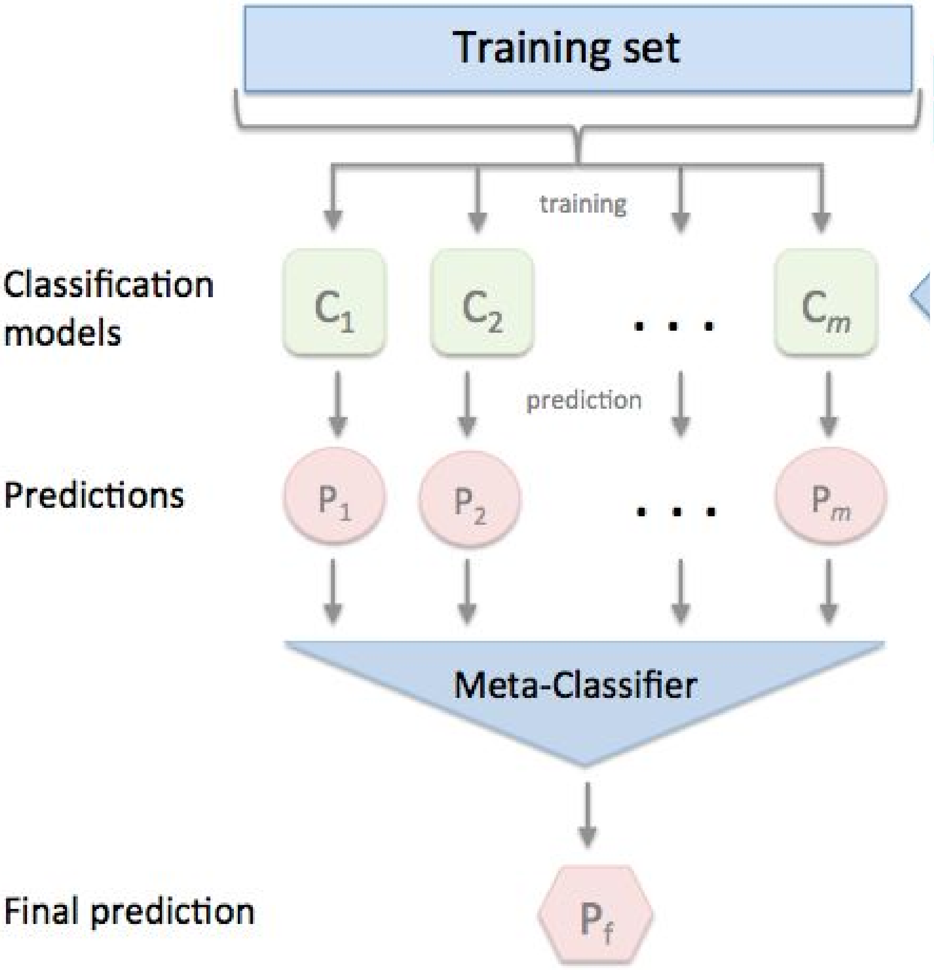
```
from xgboost import XGBClassifier

xgb_clf = XGBClassifier(n_estimators=100, max_depth=7)
xgb_clf.fit(X_train, y_train)
xgb_clf.predict(X_test)
```

# Ensemble methods -> Stacking

PyData San Luis 2017

- Involves training a learning algorithm to **combine the predictions** of several other learning algorithms.
- First, all of the other algorithms are trained using the **available data**.
- A combiner algorithm (**meta-classifier**) is trained to make a final prediction using all the predictions (**meta-features**) of the base models as inputs.

# Stacking



Training set
training
Classification models
$C_1$
$C_2$
. . .
$C_m$
prediction
Predictions
$P_1$
$P_2$
. . .
$P_m$
Meta-Classifier
Final prediction
$P_f$

# Implementation -> mlxtend

- Stacked **Classification** and Stacked **Regression.**
- Allows to operate on different **feature subsets.**
- Open Source.
- Scikit-learn interface and interaction.

```
from mlxtend.classifier import StackingClassifier

clf1 = KNeighborsClassifier(n_neighbors=1)
clf2 = RandomForestClassifier(random_state=1)
clf3 = GaussianNB()
lr = LogisticRegression()
stack_clf = StackingClassifier(classifiers=[clf1, clf2, clf3],
                          meta_classifier=lr)
```

# Ensemble Comparison table

| | Bagging | Boosting | Stacking |
|---|---|---|---|
| **Train the model** | Parallel | Sequence | By layers |
| **Dataset for each base model** | Random subset with replacement | Subset with misclassified points | Full dataset |
| **Features** | Subset | Subset | All or subset |
| **Predictions** | Committe - average | weighted majority | Meta-classifier |

# Example use case

Predict if an item is new or used using variables like title, price and category.

**Model's performance comparison**

| **Model** | **Accuracy** |
|---|---|
| Decision Tree | 76% |
| Random Forest | 81% |
| Gradient Boosting Trees | 83% |
| Stacked model | 85% |

# References


- Pattern Recognition and Machine Learning, Christopher Bishop .
- XGBoost: A Scalable Tree Boosting System, by Tianqi Chen
- Greedy Function Approximation: A Gradient Boosting Machine, by Friedman.
- [https://es.slideshare.net/katatunix/gradient-boosting](https://es.slideshare.net/katatunix/gradient-boosting)
- Mlxtend: http://rasbt.github.io/mlxtend/
- Tang, J., S. Alelyani, and H. Liu. "Data Classification: Algorithms and Applications." Data Mining and Knowledge Discovery Series, CRC Press (2015): pp. 498-500.
- Wolpert, David H. "Stacked generalization." Neural networks 5.2 (1992): 241-259.