
# Introduction to Machine Learning Homework 3 Announcement

Presenter: TA Jui-Che (Ben)

Lastest update: 2023/11/14 13:30

# Homework 3

- Deadline: 23:59, Nov. 28th (Tue), 2023
- Coding (50%): Implement ensemble methods by only using **numpy.**
  - Part 1: Decision Tree
  - Part 2: Adaboost
  - Submit your python file (.py).
  - Answer the questions (by screenshots) in the report (.pdf).
- Handwritten Questions (50%): Answer questions about linear classification methods.
  - Answer the questions (handwritten, typed, digital, etc.) in the report.

# Links

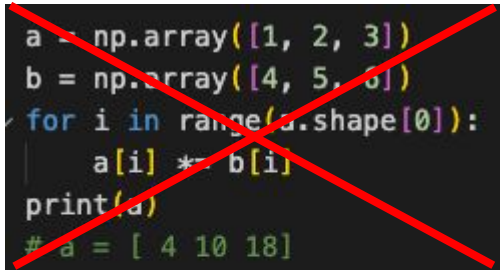
- Questions: Link
- Sample code: Link
- Dataset: Link
- Report template: Link (same as HW1)

# Environment

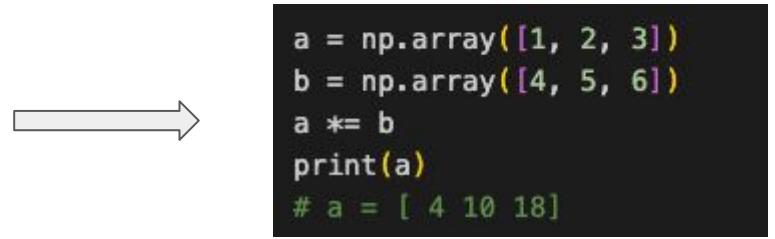
- Python version: 3.8 or newer
- Tips
  - We recommend that you use **virtual environments** when implementing your homework assignments.
  - Here are some popular virtual environment management tools:
    - Conda
    - Miniconda
    - virtualenv

# Numpy

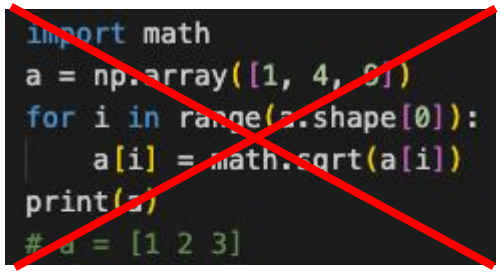
- Build-in array operations.
- Numpy Tutorial: [Link]



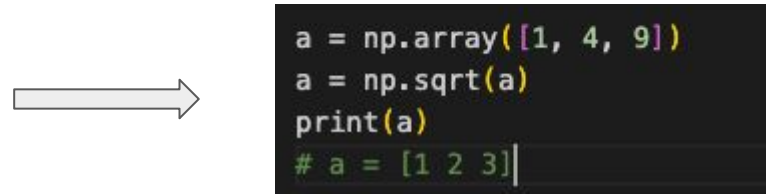
```
a = np.array([1, 2, 3])
b = np.array([4, 5, 6])
for i in range(a.shape[0]):
    a[i] *= b[i]
print(a)
# a = [ 4 10 18]
```



```
a = np.array([1, 2, 3])
b = np.array([4, 5, 6])
a *= b
print(a)
# a = [ 4 10 18]
```



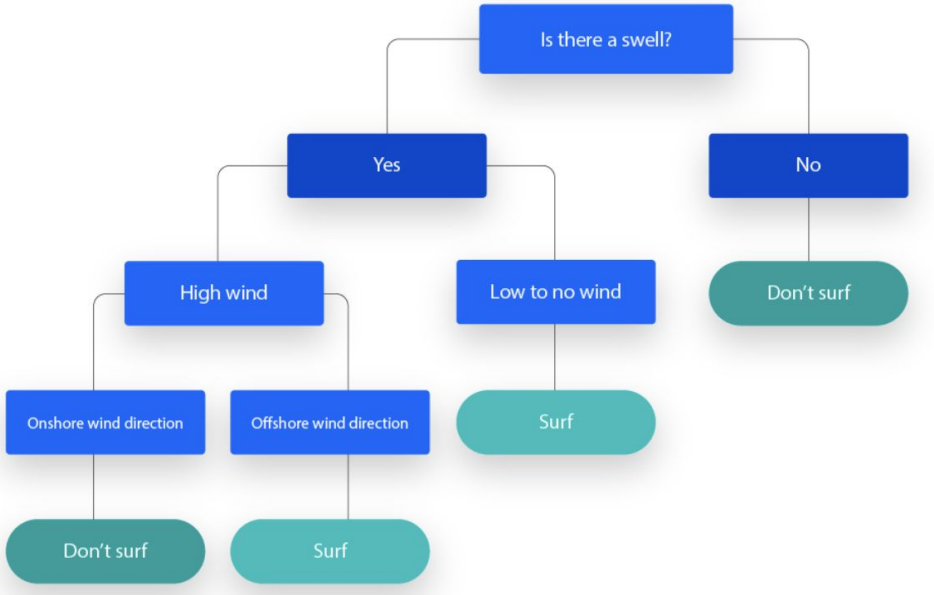
```
import math
a = np.array([1, 4, 9])
for i in range(a.shape[0]):
    a[i] = math.sqrt(a[i])
print(a)
# a = [1 2 3]
```



```
a = np.array([1, 4, 9])
a = np.sqrt(a)
print(a)
# a = [1 2 3]
```

# Decision Tree

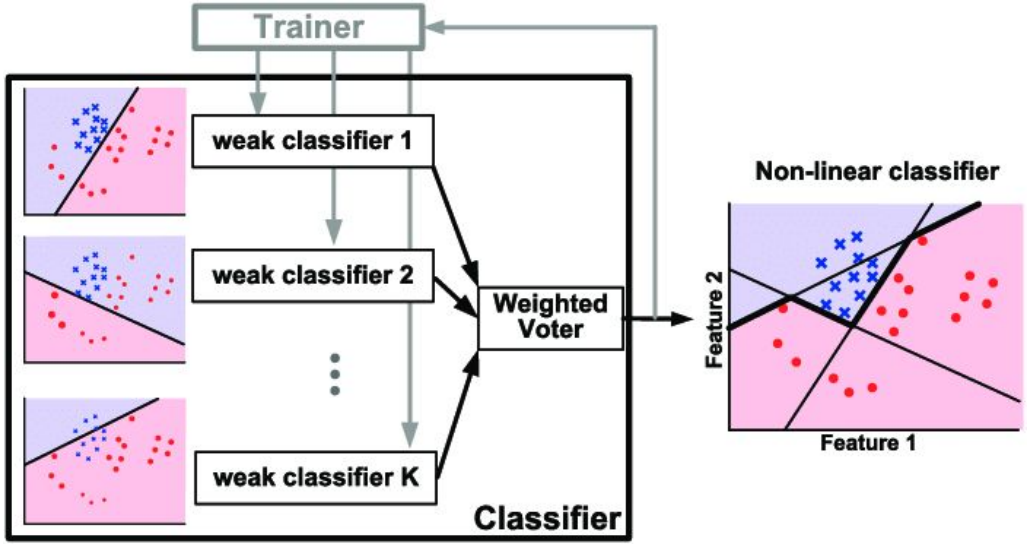
- Decision tree is a non-parametric supervised learning algorithm which has a hierarchical, tree structure, which consists of a root node, branches, internal nodes and leaf nodes.




https://www.ibm.com/topics/decision-trees

# Adaboost

- AdaBoost is a boosting technique used as an ensemble method in machine learning. It is called Adaptive Boosting as the weights are re-assigned to each instance, with higher weights assigned to incorrectly classified instances.






https://www.researchgate.net/figure/Illustration-of-AdaBoost-algorithm-for-creating-a-strong-classifier-based-on-multiple_fig9_288699540

# Dataset

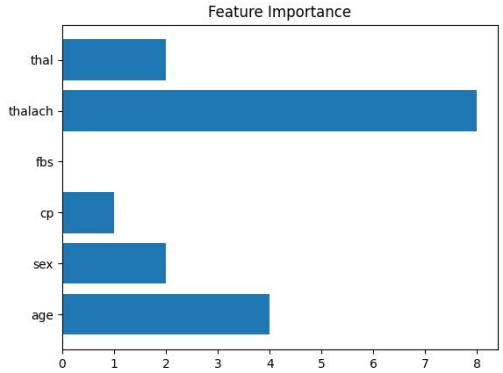
- Heart Attack Dataset
- Features
  - age
  - sex
  - cp: chest pain type (4 values)
  - fbs: fasting blood sugar > 120 mg/dl
  - thalach: maximum heart rate achieved
  - thal: 0 = normal; 1 = fixed defect; 2 = reversable defect
- Target
  - target (0 = no heart attack, 1 = heart attack)

# Decision Tree

- Requirements:
  - Implement gini index and entropy for measuring the best split of the data.
  - Implement the decision tree classifier with the following two arguments:
    - **criterion**: The function to measure the quality of a split of the data.
    - **max_depth**: The maximum depth of the tree.
- Tips
  - Your model should produce the same results when rebuilt with the same arguments.
  - You can use the recursive method to build the nodes.

# Decision Tree

- Criteria:
  - (5%) Compute the gini index and the entropy of the array [0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 1].
  - (10%) Show the accuracy score of the testing data using criterion='gini' and max_depth=7.
  - (10%) Show the accuracy score of the testing data using criterion=entropy and max_depth=7.
  - (5%) Train your model using criterion='gini', max_depth=15. Plot the feature importance of your decision tree model by simply counting the number of times each feature is used to split the data.



# Adaboost

- Requirements:
  - Implement the Adaboost algorithm by using the decision tree classifier (max_depth=1) you just implemented as the weak classifier.
  - The Adaboost model should include the following two arguments:
    - **criterion**: The function to measure the quality of a split of the data. Your model should support "gini" and "entropy".
    - **n_estimators**: The total number of weak classifiers.
- Tips
  - You can set any random seed to make your result reproducible.

# Adaboost

- Criteria:
  - (20%) Tune the arguments of AdaBoost to achieve higher accuracy than your Decision Trees.

| Points | Testing Accuracy |
|---|---|
| **20 points** | 0.8 <= acc |
| **15 points** | 0.78 <= acc < 0.8 |
| **10 points** | 0.76 <= acc < 0.78 |
| **5 points** | **0.74 <= acc < 0.76** |
| **0 points** | **acc < 0.74** |

# Code Output

- Do not modify the main function architecture.
- Your code output will look like this:

```
Part 1: Decision Tree
gini of [0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 1]:
entropy of [0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 1]:
Accuracy (gini with max_depth=7): 0.7049180327868853
Accuracy (entropy with max_depth=7): 0.7213114754098361
Part 2: AdaBoost
Accuracy: 0.8032786885245902
```

# Report

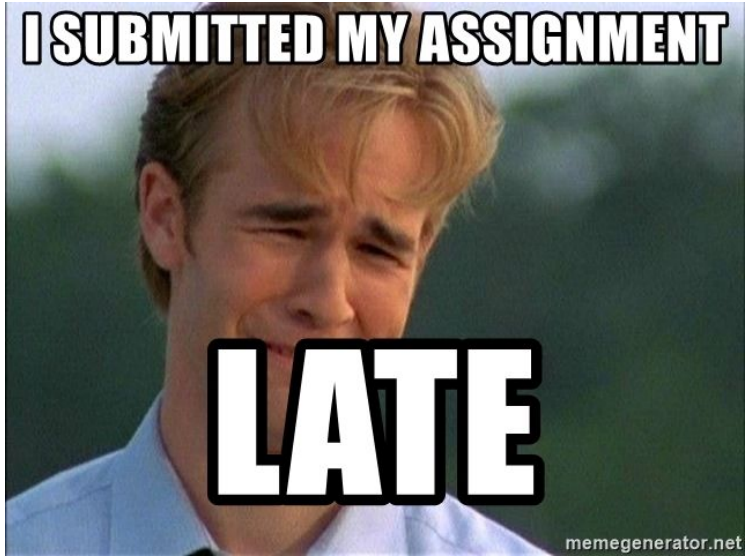
- Please follow the same report template format just like HW1.
- Link

# Submission

- Compress your code and report into a **.zip file** and submit it on E3.
- <STUDENT ID>_HW3.zip
  - <STUDENT ID>_HW3.py
  - <STUDENT ID>_HW3.pdf (do not submit .doc, .docx or others format)

# Late policy

- We will deduct a late penalty of 20 points per additional late day.
- For example, If you get 90 points but delay for two days, your will get only 50 points!



# Have Fun!


https://www.pinterest.com/pin/409475791116140835/