Bootcamp
Machine Learning

Bootcamp ML

Day04 - Decision Tree

This day aims to present a different kind of ML algorithm: decision trees.
Decision trees are historically quite popular as it is easy to demonstrate the thought process of the algorithm when
making a prediction. They are also the base of more complex algorithms (random trees...) that have very good performances.

If you don't have the time to recode everything try to at least grasps the key concepts to be able to use them wisely in the future.

Notions of the day

Gini impurity, entropy, information gain.

- Gini:

https://victorzhou.com/blog/gini-impurity/
- Entropy:

https://www.youtube.com/watch?v=9r7FIXEAGvs&feature;=youtu.be

Decision trees overview

- Statquest:
https://www.youtube.com/watch?v=7VeUPuFGJHk&list;=PLblh5JKOoLUICTaGLRoHQDuF_7q2GfuJF&index;=34
- Building a decision tree step by step:
https://www.youtube.com/watch?v=LDRbO9a6XPU&feature;=youtu.be
- Sklearn doc:
https://scikit-learn.org/stable/modules/tree.html#tree
- On decision tree regressors:
https://gdcoder.com/decision-tree-regressor-explained-in-depth/

Digging deeper

• ML course on decision trees. Beware principles are the same but implementation algorithm is not sklearn's one.
https://www.youtube.com/watch?v=eKD5gxPPeY0&list;=PLBv09BD7ez_4temBw7vLA19p3tdQH6FYO
• Nice article. Not everything relevant for the day.
https://mlcourse.ai/articles/topic3-dt-knn/
• SO on difference between several decision trees algorithms.
https://stackoverflow.com/questions/9979461/different-decision-tree-algorithms-with-comparison-of-complexity-or-performance

General rules

• The version of Python to use is 3.7, you can check the version of Python with the following command: python -V
• The norm: during this bootcamp you will follow the Pep8 standards
https://www.python.org/dev/peps/pep-0008/
• The function eval is never allowed.
• The exercices are ordered from the easiest to the hardest.
• Your exercises are going to be evaluated by someone else, so make sure that your variable names and function names are appropriate and civil.
• Your manual is the internet.
• You can also ask question in the dedicated channel in Slack: 42-ai.slack.com.
• If you find any issue or mistakes in the subject please create an issue on our dedicated repository on Github: https://github.com/42-AI/bootcamp_machine-learning/issues.

Helper

Ensure that you have the right Python version.

```
> which python
/goinfre/miniconda/bin/python
> python -V
Python 3.7.*
> which pip
/goinfre/miniconda/bin/pip
```

Exercices

Exercise 00 - Shannon's Entropy

Exercise 01 - Gini Impurity

Exercise 02 - Information Gain

Exercise 00 - Shannon's Entropy

Turn-in directory :	ex00
Files to turn in :	entropy.py
Forbidden modules :	sklearn
Forbidden functions :	Anything that computes directly the functions.
Remarks :	Read the doc

Objective:
You must implement the following formula as a function:

$$S(X) = -\sum_{i=1}^{C} p_i \log_2(p_i)$$

Where

- X is a vector of dimension N * 1
- C is the number of different class found in X's components
- p_i is the probability of class i in X: $p_i = \frac{n}{N}$ where n is the number of components of X belonging to class i
- log_2 is the logarithm function in base 2

Instruction:
In the entropy.py file, create the function as per the instructions given below:

```
def entropy(array):
    """
    Computes the Shannon Entropy of a non-empty numpy.ndarray
    :param numpy.ndarray array:
    :return float: shannon's entropy as a float or None if input is not a
non-empty numpy.ndarray
    """
```

Output examples:

```
Shannon entropy for [] is None
Shannon entropy for {1, 2} is None
Shannon entropy for bob is None
Shannon entropy for [0 0 0 0 0 0] is 0.0
Shannon entropy for [6] is 0.0
Shannon entropy for ['a' 'a' 'b' 'b'] is 1.0
Shannon entropy for ['0' '0' '1' '0' 'bob' '1'] is 1.4591479170272448
Shannon entropy for [0 0 1 0 2 1] is 1.4591479170272448
Shannon entropy for ['0' 'bob' '1'] is 1.584962500721156
Shannon entropy for [1. 1. 1. 1. 1. 1. 1. 1. 1. 1.] is 0.0
Shannon entropy for [0. 1. 1. 1. 1. 1. 1. 1. 1. 1.] is 0.4689955935892812
Shannon entropy for [0 0 1] is 0.9182958340544896
```

Exercise 01 - Gini Impurity

Turn-in directory :	ex01
Files to turn in :	gini.py
Forbidden modules :	sklearn
Forbidden functions :	Anything that computes directly the functions.
Remarks :	Read the doc

Objective:
You must implement the following formula as a function :

$$G(X) = 1 - \sum_{i=1}^{C} p_i^2$$

Where

- X is a vector of dimension N * 1
- C is the number of different classes found among X's components
- p_i is the probability of class i in X: $p_i = \frac{n}{N}$ where n is the number of components of X belonging to class i

Instruction:
In the entropy.py file, create the function as per the instructions given below:

```
def gini(array):
    """
    Computes the gini impurity of a non-empty numpy.ndarray
    :param numpy.ndarray array:
    :return float: gini_impurity as a float or None if input is not a
non-empty numpy.ndarray
    """
```

Output examples:

```
Gini impurity for [] is None
Gini impurity for {1, 2} is None
Gini impurity for bob is None
Gini impurity for [0 0 0 0 0 0] is 0.0
Gini impurity for [6] is 0.0
Gini impurity for ['a' 'a' 'b' 'b'] is 0.5
Gini impurity for ['0' '0' '1' '0' 'bob' '1'] is 0.6111111111111112
Gini impurity for [0 0 1 0 2 1] is 0.6111111111111112
Gini impurity for ['0' 'bob' '1'] is 0.6666666666666667
Gini impurity for [1. 1. 1. 1. 1. 1. 1. 1. 1. 1.] is 0.0
Gini impurity for [0. 1. 1. 1. 1. 1. 1. 1. 1. 1.] is 0.18
Gini impurity for [0 0 1] is 0.4444444444444445
```

Exercise 02 - Information Gain

Turn-in directory :	ex02
Files to turn in :	information_gain.py
Forbidden modules :	sklearn
Forbidden functions :	Anything that computes directly the functions.
Remarks :	Read the doc

Objective:
You must implement the following formula as a function:

$$IG(q) = S_0 - \sum_{i=1}^{q} \frac{n_i}{N} S_i$$

Where

- X is a vector of dimension N * 1
- S_0 is the actual value of impurity (or entropy)
- q is the number of different parts of X found after the split
- n_i is the number of components of part i
- S_i is the impurity (or entropy) value of part i

Instruction:
In the information_gain.py file, create the function as per the instructions given below:

```
def information_gain(array_source, array_children_list, criterion='gini'):
    """
    Computes the information gain between the first and second array using
the criterion ('gini' or 'entropy')

    :param numpy.ndarray array_source:
    :param list array_children_list: list of numpy.ndarray
    :param str criterion: Should be in ['gini', 'entropy']
    :return float: Shannon entropy as a float or None if input is not a
non-empty numpy.ndarray or None if invalid input
    """
```

Output examples:

```
Information gain between [] and [1. 1. 1. 1. 1. 1. 1. 1. 1. 1.] is None with
criterion 'gini' and None with criterion 'entropy'
Information gain between ['a' 'a' 'b' 'b'] and {1, 2} is None with criterion
'gini' and None with criterion 'entropy'
Information gain between [0. 1. 1. 1. 1. 1. 1. 1. 1. 1.] and [1. 1. 1. 1. 1.
1. 1. 1. 1. 1.] is 0.18 with criterion 'gini' and 0.4689955935892812 with
criterion 'entropy'
Information gain between ['0' '0' '1' '0' 'bob' '1'] and [array(['0', 'bob',
'1'], dtype='Information gain between ['0' '0' '1' '0' 'bob' '1'] and [0 0 1
0 2 1] is 0.0 with criterion 'gini' and 0.0 with criterion 'entropy'
```

Exercise 03 - Understanding the Mathematical Concepts

Turn-in directory :	ex03
Files to turn in :	answers.txt
Forbidden modules :	NA
Forbidden functions :	NA
Remarks :	Read the doc

Objective:

The aim of this exercise is to check your understanding of the three key mathematical concepts for decision trees:

- Gini impurity of a dataset.
- Shannon entropy of a dataset.
- Information gain between two datasets.

Instruction:

In the answers.txt file, answer the following questions in 3 sentences maximum. The idea is to understand the underlying concepts. These are simple questions (no traps!)

- Define what Gini impurity is about and what it measures. (No mathematical formula, just the general concept in words).
- Define what Shannon entropy is and what it measures. (No mathematical formula, just the general concept in words).
- Define what Information gain is and what it measures. (No mathematical formula, just the general concept in words).
- Explain how these 3 concepts are used for decision trees.
- If the dataset has 2 classes, explain what are the boundaries (minimum and maximum) of Gini impurity and Shannon entropy.
- What does it mean if the Gini impurity is 0? If Shannon entropy is 0 ?

Bonus questions

7) Why should you use Gini impurity as default?

8) What are the pros and cons of just using the criteria Information Gain positive or negative to pick the feature of a decision tree? Can we mitigate this risk ?

Exercise 04 - Understanding better Decision Tree for Classification

Turn-in directory :	ex04
Files to turn in :	answers.txt
Forbidden modules :	NA
Forbidden functions :	NA
Remarks :	Read the doc

Objective:

The aim is to understand some key points of decision trees.

Instruction:

In the answers.txt file, answer the following questions in 3 sentences maximum. The idea is to understand the underlying concepts. These are simple questions (no traps!)

- What are the pros of using decision trees?
- What are the cons of using decision trees?
- What is overfitting? How does it apply to decision trees?
- What can be done to avoid overfitting in decision trees?
- What is the name of the algorithm used by sklearn for classification decision trees?

Exercise 05 - Code the DecisionTreeClassifier Fit Function

Turn-in directory :	ex05
Files to turn in :	decision_tree_classifier.py
Forbidden modules :	sklearn
Forbidden functions :	Anything that makes the coffee for you.
Remarks :	Read the doc

Objective:

The aim of this exercise is to code a simplified version of Sklearn's DecisionTreeClassifier. This part focuses on the fit function, which trains a Decision tree on a given dataset.

Instruction:
Fill in the DecisionTreeClassifier class. You can import the Node class from the node.py file. Feel free to use it, amend it or not.
Please note that depending on a few implementation choices, the output can be different. As you may have seen decision trees
can have a high variance. Try to figure out why.
Feel free to add other methods and functions that you write (you can reuse the information_gain function that you already created).

```
import pandas as pd

from node import Node


class DecisionTreeClassifier:
    def __init__(self, criterion='gini', max_depth=10):
        """

        :param str criterion: 'gini' or 'entropy'
        :param max_depth: max_depth of the tree (Decision tree creation
stops splitting a node if node.depth >= max_depth)
        """
        self.root = None # Root node of the tree
        # Your code here. You can add more things if needed

    def fit(self, X, y):
        """
        Build the decision tree from the training set (X, y). The training
set has m data_points (examples).
        Each of them has n features.

        :param pandas.Dataframe X: Training input (m x n)
        :param pandas.Dataframe y: Labels (m x 1)
        :return object self: Trained tree
        """
        # Your code here. You can add more things if needed


if __name__ == '__main__':
    from sklearn.model_selection import train_test_split
    from sklearn.datasets import load_iris
    # sklearn is not allowed in the classes.

    # Test on iris dataset
    iris = load_iris()
    X = pd.DataFrame(iris.data)
    y = pd.DataFrame(iris.target)
    X_train, X_test, y_train, y_test = train_test_split(X, y,
train_size=0.7, random_state=1)

    dec_tree = DecisionTreeClassifier()
    dec_tree.fit(X_train, y_train)
    root = dec_tree.root
    print("TEST ON IRIS DATASET")
    print("Root split info = 'Feature_{}{}{}'\n".format(root.split_feature,
root.split_kind, root.split_criteria))
    print("5 first lines of the labels of the left child of root
=\n{}\n".format(root.left_child.y.head()))
    print("5 first lines of the labels of the right child of root
=\n{}".format(root.right_child.y.head()))
```

Output examples:

```
TEST ON IRIS DATASET
Root split info = 'Feature_2 <= 1.9'

5 first lines of the labels of the left child of root =
    0
18 0
4 0
45 0
39 0
36 0

5 first lines of the labels of the right child of root =
     0
118 2
59 1
117 2
139 2
107 2
```

Exercise 06 - Code the DecisionTreeClassifier Prediction Functions

Turn-in directory :	ex06
Files to turn in :	decision_tree_classifier.py
Forbidden modules :	sklearn
Forbidden functions :	Anything that makes the coffee for you.
Remarks :	Read the doc

Objective:

The aim of this exercise is to code a simplified version of Sklearn's DecisionTreeClassifier.
This part focuses on expanding the previous exercise's class in order to add predicion functions (predict_proba and predict).
The method score aims to get the accuracy of the trained tree on a given dataset.

Instruction:
Complete the previous Decision Tree classifier.
Please note that depending on a few implementation choices, the output can be different. As you may have seen decision trees can have a high variance. Try to figure out why.
Feel free to add other methods and functions that you write.

```
class DecisionTreeClassifier:
    # Use the classes created previously and complete the
DecisionTreeClassifier class with these 3 methods
    def predict_proba(self, X):
        """
        Predict class probabilities of the input samples X.

        :param pandas.Dataframe X: Input samples. (m_samples, n_features)
        :return pandas.Dataframe predicted_probas: array of shape
(m_samples, c_classes)
        """

    def predict(self, X):
        """
        Predict class for input samples X

        :param pandas.Dataframe X: Input samples. (m_samples, n_features)
        :return pandas.Series predicted_class: Predicted classes. (m_samples
x 1)
        """

    def score(self, X_test, y_test):
        """
        Get accuracy.

        :param pandas.Dataframe X_test: Input samples. (m_samples,
n_features)
        :param pandas.Series y_test: Predicted classes. (m_samples x 1)
        :return float accuracy: Accuracy of the classifier on the validation
set given.
        """

if __name__ == '__main__':
    from sklearn.model_selection import train_test_split
    from sklearn.datasets import load_wine
    # sklearn is not allowed in the classes.


    # Test on wine dataset
    wine = load_wine()
    X = pd.DataFrame(wine.data)
    y = pd.DataFrame(wine.target)
    X_train, X_test, y_train, y_test = train_test_split(X, y,
train_size=0.7, random_state=1)

    dec_tree = DecisionTreeClassifier(max_depth=2)
    dec_tree.fit(X_train, y_train)
    predict_probas = dec_tree.predict_proba(X_test)
    accuracy = dec_tree.score(X_test, y_test[0])
    print("TEST ON WINE DATASET")
    print("5 first lines of data input =\n{}\n".format(X.head()))
    print("5 first lines of predictions probablilities per class
=\n{}".format(predict_probas.head()))
    print("Accuracy for max_depth=2: {}\n".format(accuracy))
```

```
    dec_tree = DecisionTreeClassifier(max_depth=10)
    dec_tree.fit(X_train, y_train)
    predict_classes = dec_tree.predict(X_test)
    accuracy = dec_tree.score(X_test, y_test[0])
    print("5 first lines of predictions classes
=\n{}".format(predict_classes.head()))
    print("Accuracy for max_depth=10: {}".format(accuracy))
```

Output examples:

```
TEST ON WINE DATASET
5 first lines of data input =
     0     1     2     3     4     5     6     7     8     9    10    11
12
0 14.23 1.71 2.43 15.6 127.0 2.80 3.06 0.28 2.29 5.64 1.04 3.92 1065.0
1 13.20 1.78 2.14 11.2 100.0 2.65 2.76 0.26 1.28 4.38 1.05 3.40 1050.0
2 13.16 2.36 2.67 18.6 101.0 2.80 3.24 0.30 2.81 5.68 1.03 3.17 1185.0
3 14.37 1.95 2.50 16.8 113.0 3.85 3.49 0.24 2.18 7.80 0.86 3.45 1480.0
4 13.24 2.59 2.87 21.0 118.0 2.80 2.69 0.39 1.82 4.32 1.04 2.93 735.0

5 first lines of predictions probablilities per class =
            0         1         2
161 0.000000 0.000000 1.000000
117 0.017857 0.875000 0.107143
19 0.921053 0.078947 0.000000
69 0.017857 0.875000 0.107143
53 0.921053 0.078947 0.000000
Accuracy for max_depth=2: 0.8703703703703703

5 first lines of predictions classes =
161    2
117    1
19     0
69     1
53     0
dtype: int64
Accuracy for max_depth=10: 0.9259259259259259
```

Exercise 07 - Using Decision Trees for Regression

Turn-in directory :	ex07
Files to turn in :	answers.txt
Forbidden modules :	NA
Forbidden functions :	NA
Remarks :	Read the doc

Objective:

The aim is to understand some key points of decision trees for regression.

Instruction:

In the answers.txt file, answer the following questions in 3 sentences maximum. The idea is to understand the underlying concepts. These are simple questions (no traps!)

- How do the decision trees for regression work?
- Is it still appropriate to use Shannon's entropy or Gini's coefficient? If not, what measure do we use instead, when dealing with regression?
- For a regression task, when is it better to use a decision tree than linear regression?