Software Engineering

WS 2022/23, Assignment 04

Prof. Dr. Sven Apel
Annabelle Bergum
Sebastian Böhm
Christian Hechtl

Handout: 16.01.2023
Handin: 30.01.2023 23:59 CET

Organizational Section:

- The assignment must be accomplished by yourself. You are not allowed to collaborate with anyone. Plagiarism leads to failing the assignment.
- The deadline for the submission is fixed. A late submission leads to a desk reject of the assignment.
- The submission must consist of a ".ipynb" file fulfilling the following criteria:
  - You **used the Jupyter Notebook template** provided in the materials section on the course's CMS page.
  - Your name and matriculation number are included as specified by the template.
  - Do **not change the signatures of the methods provided**, this will make your submission invalid.
  - Do **not use any further libraries or imports**.
  - Make sure the given test cases run on your implementation.
- Any violation of these submission format rules leads to a desk reject of the assignment.
- Questions regarding the assignment can be asked in the forum or during tutorial sessions. Please do not share any parts that are specific to your solution, as we will have to count that as attempted plagiarism.
- If you encounter any technical issues, inform us immediately.

General Information

In this assignment, we analyze a non-functional property, performance, of a fictional tool "SECompress", a configurable command-line tool for compressing data. In addition, the data can be encrypted, signed, segmented, and/or time-stamped. All this functionality is modeled by the feature diagram in Figure 1. For all the tasks in this Assignment we will use this tool with the features given in this diagram.

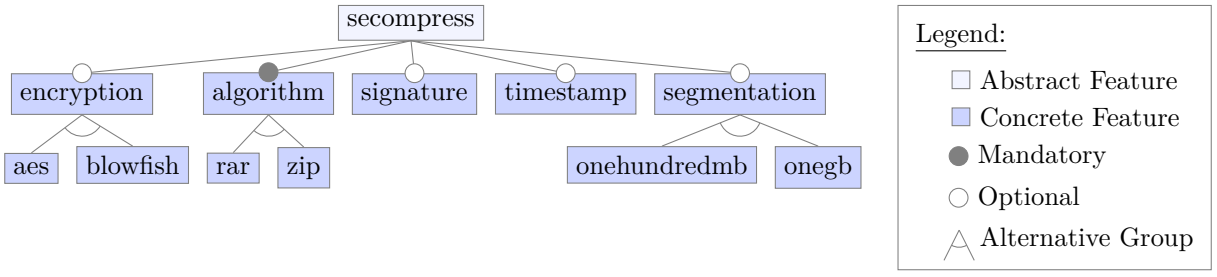

Figure 1: The Feature Diagram of "SECompress"

Jupyter Notebook You have to implement your solution using the provided Jupyter Notebook "Template Assignment 04.ipynb" that can be downloaded from the CMS.

- This Jupyter Notebook contains comments where your implementation goes, marked with "TODO"
- It contains example test cases. For the tests, you need to download the "*.csv" files from the material section in the CMS and save them in the same folder.
- No additional imports are allowed, otherwise your submission is invalid and leads to a desk reject.
- No additional libraries are allowed, otherwise your submission is invalid and leads to a desk reject.

- No changing of signatures is allowed, otherwise your submission is invalid and leads to a desk reject.

To use the Jupyter Notebook Template you need to install Juypter. You can use the install guideline `https://jupyter.org/install` of Jupyter, but on Windows we recommend using Anaconda to use Jupyter `https://docs.anaconda.com/ae-notebooks/user-guide/basic-tasks/apps/jupyter/index.html`. If you encounter any problems try googling them, there are lots of tutorials and forum discussions already online. Otherwise you can use the SE forum, but please try to google it on your own first.

CART Data Structure For this assignment sheet we use the following internal datastructure for a CART. We represent a CART as a python dict with exactly the entries as shown in the example below. This example CART has three nodes. The root node "X", and the two child nodes "XL" and "XR". As you can see the child nodes only have a name and a mean but all other fields are set to None. A parent node also has a name and a mean but additionally a feature by which the split is performed, the error of the split and two sucessors.

```
cart = { "name":"X", "mean":456, "split_by_feature": "aes", "error_of_split":73,
        "successor_left": { "name":"XL", "mean":1234, "split_by_feature": None,
        ↪  "error_of_split":None,
               "successor_left":None,
               "successor_right":None
        },
        "successor_right":{ "name":"XR", "mean":258, "split_by_feature": None,
        ↪  "error_of_split":None,
               "successor_left":None,
               "successor_right":None
        }
}
```

It is important to use exaclty these names for the features:

```
features = ["secompress", "encryption", "aes", "blowfish", "algorithm", "rar", "zip",
↪  "signature", "timestamp", "segmentation", "onehundredmb", "onegb"]
```

Performance Data The performance data, given in a csv file, contains different configurations of SECompress with performance measurements, in the same format you saw in the lecture (Slide 26).

```
Id,secompress,encryption,aes,blowfish,algorithm,rar,zip,signature,timestamp,segmentation,
onehundredmb,onegb,performance
0,1,0,0,0,1,1,0,0,0,0,0,0,750
1,1,0,0,0,1,1,0,0,0,1,1,0,773
2,1,0,0,0,1,1,0,0,0,1,0,1,770
3,1,0,0,0,1,1,0,0,1,0,0,0,750
4,1,0,0,0,1,1,0,0,1,1,1,0,773
```

Task 1 [15 Points]

a) Implement an algorithm that creates a CART from performance data as presented in the lecture. The CART must be implemented by yourself, you must not take a prefabricated Python implementation. The format of the sample data and the representation of the CART are described below. If two split options are equally good, we use the alphabetic ordering of feature names as tie breaker.

You can use the File from the CMS "Performance_01.csv" as an example test case. Then you should read this sample set csv file into a dataframe to work on this data. Now it is your turn, you must implement how we get a CART out of this dataframe, as presented in the lecture (Slides 26 to 28).

```
def get_cart(sample_set_csv):
        # The sample_set_csv is a file path to a csv data, this can be imported into a
        ↪  dataframe
        df = pd.read_csv(sample_set_csv)
        # TODO: Write your code here. And change the return.
        return { "name":"X", "mean":1234, "split_by_feature": "rar", "error_of_split":42,
        ↪                          "successor_left":None,"successor_right":None}
```

Task 2 [10 Points]

In the next task you will work on CARTs.

a) Given a CART you should determine the name of the feature with the highest influence. [2 Points]

```python
def get_highest_influence_feature(cart):
        # TODO: Write your code here. And change the return.
        return "secompress"
```

b) Given a CART and a sample set, calculate the error rate. The error rate is computed by predicting the value with the given CART for every sample in the sample set and then averaging the differences between the predicted and original performance. [4 Points]

```python
def get_error_rate(cart, sample_set_csv):
        # The sample_set_csv is a file path to a csv data, this can be imported into a
        ↪ dataframe
        df = pd.read_csv(sample_set_csv)
        # TODO: Write your code here. And change the return.
        return 42
```

c) Given a CART and a partial configuration, compute a complete configuration which is in line with the partial configuration and has the least predicted costs.
The partial configuration and the configuration are python sets. [4 Points]

```python
def get_optimal_configuration(cart, partial_configuration):
        # TODO: Write your code here. And change the return.
        return {"rar", "timestamp"}
```