Code documentation of Python rFactor project

January 2024

1. A few words about project from author

The project was started for several reasons. Main reason was to connect my hobby with something practical. As videogames-lover I love, as much as playing them, editing and modifying their files. Personally, I love racing games, but I always saw missing functionalities like real career mode for user. If I play Formula 1 or MotoGP, I always wanted to see progress like changing drivers skills, drivers transfers after a season, liveries changed, standings for drivers and teams and many more.

I decided to put additional functionalities in Python in external file. User manages only two things: **GUI** to starts new season and **Excel input/output file**. I chose rFactor videogame as its files are very easy to access and modify (files are able to be opened with notepad, so **text-scraping** methods could be used).

The aim was to add functionalities like:

a) changing drivers between teams based on their results (available in saved game files) and **reading them**
b) updating their skills based on their results and age using **mathematical formulas and numpy functions**,
c) easy assignment of drivers and teams built on automatical assignment to created **Python classes** Driver and Team,

d) **machine learning methods** used to check if driver should leave the team (perceptron) and proper assignment to the team (linear regression),
e) models built on MotoGP data from Wikipedia with simple **web-scraping** methods,
f) some simple visualizations added in machine learning section with support of **matplotlib.pyplot** package,
g) and taking care of code to assign as many functionalities as separate **functions** and many transformations done by **pandas** transformations to handle with tones of DataFrames.

Code was created in July-September 2023. To run functionalities, it is required rFactor 2 game installed and all paths assigned in Excel input/output file. Whole code and all ideas contained in this project belong to me (github: igorsz04).

2. Rfactor_functions

Python file containing main functions used by other parts of project. It starts from assigning variables to Excel input file (team_info.xlsm). Variables take data like year, series name, user's name, paths to files of game. They are needed to correctly use program for further purposes.

Functions assigned in this part:

1) copying_orig_files – creating a backup for original files
2) find_name – created to gain names from xml files of videogame

3) create_table – creating full season table with drivers, teams and cars from single DataFrames containing results of single race
4) create_team_table – similar to “create_table”, used to create team standings
5) create_table_positions – creating table with positions of riders used for future purposes
6) set_new_season – function used to create a new season (automatically update team_info file) and all standings from zero
7) loop_riders_grid – information about starting positions used by another loop for another table (about grid position changes)

Additionally, some functionalities were added to filter races with specific series name in order to distinguish races in videogame which should be used for our standings from other, random ones. Points table is added too to provide number of points for standings.

For-loops are added to check every file in a folder and then checking every race in order to add to season’s list information about race tracks, drivers in every race, tables, etc.

Drivers are loaded from two different paths. Previously mentioned, drivers were downloaded from results files, however their information is reached from skills files also. Using simple text-scraping methods their data about nationality, skills, records is gained.

3. Rfactor_class_driver

This part of code contains only class Driver which describes data of drivers, assign them new skills (old_skill vs skill) based on their achievements (results + other tables like grid position changes) using more advanced mathematical formulas or numpy functions. Class contains additional functions to show skills specific drivers and overall counted for better recognition. The most important part is in function "save_skills" which save new skills to specific input text file used by videogame by searching driver and assigning new skills. Last function could be used during finishing season and is assigned to GUI.

4. Rfactor_data_classes

This part focuses on assigning all drivers to rider_class list. Process of class objects creation is done automatically fully with names, skills and other necessary data and no driver is assigned manually. Function "new_season_riders" is used to run class Driver functionalities.

Last part focuses on creating class Team and its functionalities like assigning prestige of team (it is later used for better recognition on drivers transfer market). Additionally there are some functions team_save_info (to save teams data in Excel file) and loop to add teams automatically to team_class list.

5. Rfactor_machine_learning

First part is commented as this contains webscraping of Wikipedia. Some pages containing MotoGP results for specific seasons were web-scraped to receive data, however tables change sometimes and it is decided to download data to external file and later read with variable table_join_all. Some additional changes have been done manually to assign same team names between seasons (due to sponsorship reasons they are different) and prepare for later machine learning.

In next section there are two tables created to compare data of riders and teams between season x and season x+1 and check if rider changed team and if he was in factory team and his position. Some exceptions are done manually.

Finally, data was prepared for some machine learning functionalities. **Perceptron** method is used to decide if rider leaves team (due to too good or too bad results) or stays at the team. This model is not perfect as for some specific results it could show all riders change the team or no rider changes (many False positives and False negatives), so model cannot achieve high sensitivity or high specificity. Reality is that model would never be able to predict it well, as there are many different variables deciding about staying/leaving team. But it is done to prepare transfers after season automatically without user assignment and some restrictions were added later.

Another machine learning part is prepared with **Linear regression**. After preparing data for prediction, linear regression model predicts value of rider on market. The lower value, the better rider is. It is used based on results and it helps assigning of riders to teams at the end of season.

6. Rfactor_transfers

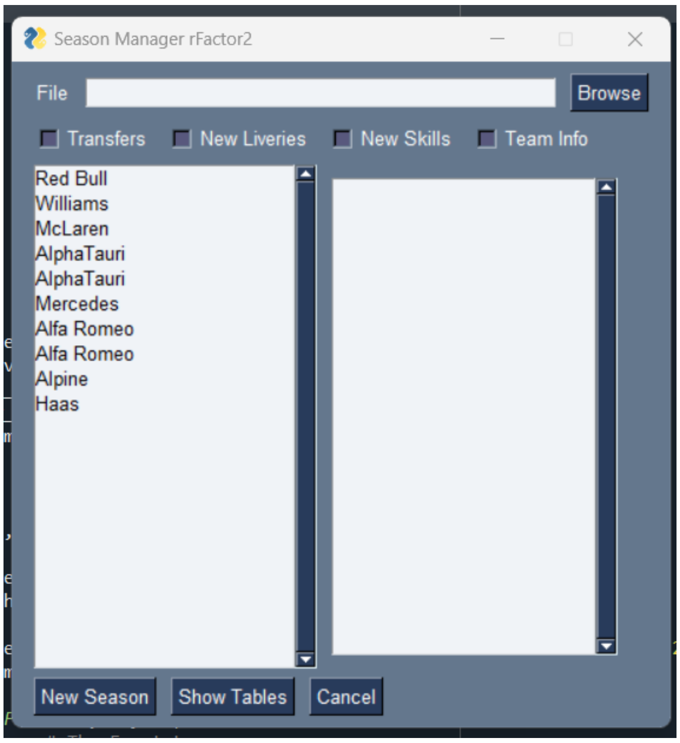
The last part of code provides main functionality for the program which are transfers. It cooperates with Excel input/output file which stores also past season standings and loads data from this. Transfers are done based on different of expectations between teams and drivers. For instance, if team was in team standings at position 5, it would require position 10 for their riders. Any difference between rider position and position 10, it builds higher probability of leaving a team by rider (too good or too bad for a team).

Section uses previous assigned machine learning functions (perceptron and linear regression) to create line-up for next year. If there is too many changes or too few, it is restricted to 25%-75% rule later (at least 25% riders change, maximum 75% riders change).

To make transfer more real, database contains also young riders which grows every year and their skills is bigger with every season automatically. On transfer market they are mixed with current drivers. Linear regression is used to set "prestige" of drivers (the lower value, the better rider) and ranking is set based on this. For teams, ranking is set by prestige taken from excel file. Then tables are merged to combine best drivers with best teams and transfer are being made. Additionally, user is allowed to change its own team manually.

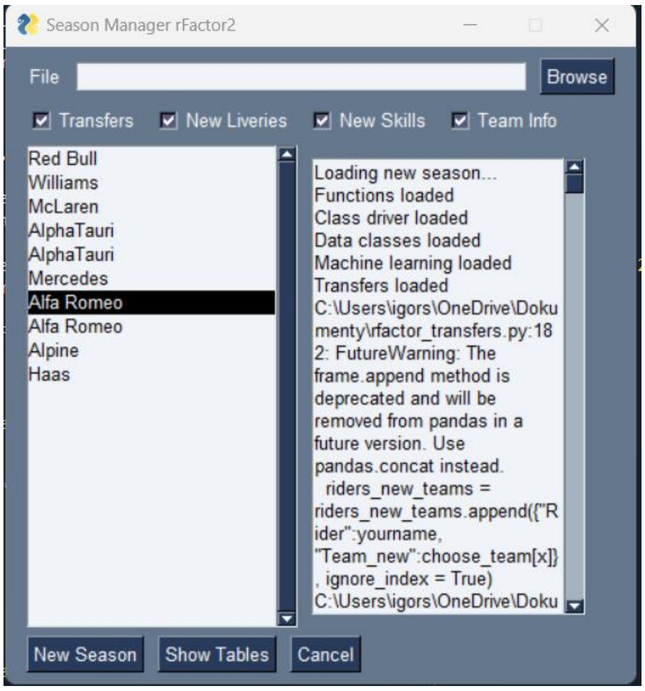
Function new_liveries is used to assign new livery for teams. As it is in real (motor)sport, teams change sometimes their sponsors after end of the season. Functionality is in testing phase and still does not work correctly + it requires additional graphic files which are not created yet.

7. Rfactor_gui



This is how GUI for project looks like. Functionalities are user-friendly. "Show tables" opens Excel input-output file. New season runs all necessary functions to finish old season and run a new one (changing year, save tables, create tables from zero). It is possible also to change a team (list of teams at the left contains teams which have free seat after end of the season). If team is shown twice (like AlphaTauri), it means both riders were set to "change" a team. User can select team for a new season.

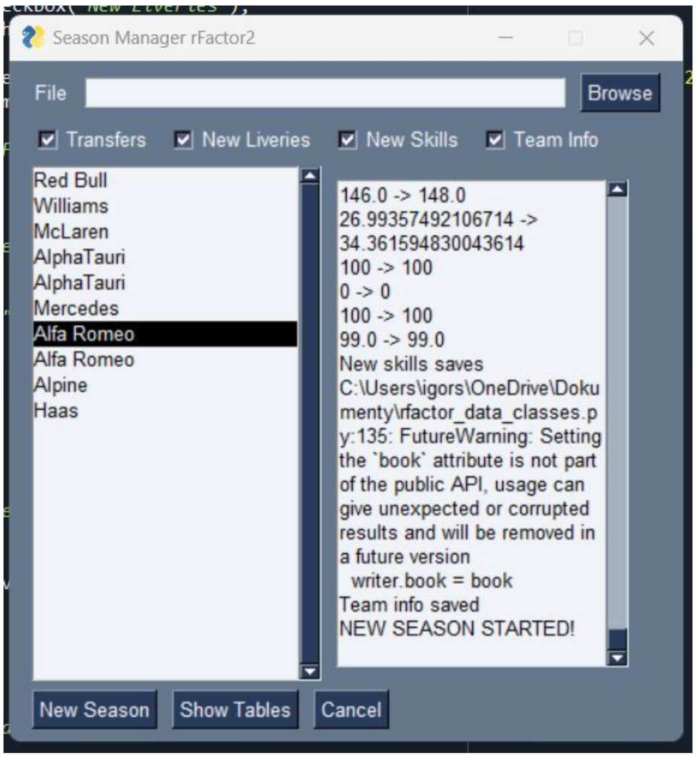
There are additional functions which can be run with "New Season" run. "Transfers" make transfers of drivers after end of the season. "New Liveries" allow user to automatically assign new liveries for teams from specific folder. "New skills" assign new skills for drivers after season (based on age, results, etc). "Team info" updates information about team (prestiges, etc). After clicking new season, all information should appear at the right. It also check if all parts are being run (functions, class driver, data classes, machine learning, transfers).



Red Bull
Williams
McLaren
AlphaTauri
AlphaTauri
Mercedes
Alfa Romeo
Alfa Romeo
Alpine
Haas

```
Loading new season...
Functions loaded
Class driver loaded
Data classes loaded
Machine learning loaded
Transfers loaded
C:\Users\igors\OneDrive\Doku
menty\rfactor_transfers.py:18
2: FutureWarning: The
frame.append method is
deprecated and will be
removed from pandas in a
future version. Use
pandas.concat instead.
  riders_new_teams =
riders_new_teams.append({"R
ider":yourname,
"Team_new":choose_team[x]}
, ignore_index = True)
C:\Users\igors\OneDrive\Doku
```

At the end of right list, there should be notification "NEW SEASON STARTED!" meaning that all functions were run and changes have been done.



Red Bull
Williams
McLaren
AlphaTauri
AlphaTauri
Mercedes
Alfa Romeo
Alfa Romeo
Alpine
Haas

```
146.0 -> 148.0
26.99357492106714 ->
34.361594830043614
100 -> 100
0 -> 0
100 -> 100
99.0 -> 99.0
New skills saves
C:\Users\igors\OneDrive\Doku
menty\rfactor_data_classes.p
y:135: FutureWarning: Setting
the `book` attribute is not part
of the public API, usage can
give unexpected or corrupted
results and will be removed in
a future version
  writer.book = book
Team info saved
NEW SEASON STARTED!
```