Add type annotations to the scipy.special submodule (could be another module)

# Rationale and Goals

Adding type hinting to all submodules will enable scipy to not only be able to have checks that can catch problems with future modelings, but also extend that ability to it's users, enabling developers to take much bigger leaps in refactoring. That creates support to assert that their changes will not create regressions and problems further down the line.

Recently type hinting support has been added to numpy on version 1.20, with important type definitions to common objects in the lib, and this will empower scipy to define which objects should be getting to which points of the code.

# Challenges

This is an interesting task in the sense that there is a lot of aspects that will be unveiled as it is developed, for example:

- Since Numpy's typing was deployed on December of 2020, it is expected that it's use is not yet mature, and maybe there will be bugs to be reported, which I could be solving or if that seems complex, finding a workaround.
- As @person142 points out in this issue: solving type hinting for scipy.special can possibly reveal another whole stack of problems, which need to be reported and dealt with
- Still talking about this issue, it says that once we explore this submodule this will empower more members of the community to help with type hinting, and I plan to find a way to document this as better as I can on places such as CONTRIBUTING.md or wherever seems fit.
- Adding type hints may be straightforward on .py files, but the scipy.special module is filled with a lot of different extensions which will all have their own unique challenges to be included on the checks (.pyf, .pyx, etc)

- Create stubs wherever needed.

# Why scipy.special?

Mostly because it was the module pointed by @person142 on github as a good case point for adopting type hints all over scipy. But it occurred to me that it could be also because it’s easy enough to be a proof of concept (and thus, too little of a task for the 12-week period of GSoC). If that’s the case we can surely move the project to a more dense module like scipy.linalg

# Timeline

The following timeline is a rough draft and needs to nail down some details before the final version.

May 17 - June 7 (Before coding period)

Community Bonding Period, getting to know the project and the people better, preparing all tools for the coming work, mapping how to work with different extensions.

June 7 - June 27

Inicial proof of concept of stubs and interactions that seem fuzzy, this is the time to understand all areas that will need careful attention along the project.

June 28 - July 25

Implement main stub dependencies and simultaneously type (typefy ?) code with its respective tests and try to look for inconsistencies between expected definitions and common place usage.

July 26 - August 23

Finish implementing all types.

# About me

I am a CS grad student from the 6th semester in Brazil, I’ve been an intern at a data science company for 18 months now, where I’ve been lucky to have awesome people orienting me on building pipelines, dashboards and occasionally messing with a model here and there.

More related to this proposal, I helped a friend of mine to develop a bit of a type hinted telegram bot api [1] [2] [3] which kickstarted me into the subject and on open source in general. Since then I’ve tried my hand with a few PR’s here and there, [1] [2] [3] but it’s still difficult to find tasks I am able to do, so to me this project represents the chance to get fluency at helping open source (: