**How to Ensure a Successful Code Review**

When you win a Zindi competition, you’ll be expected to submit both code and documentation to Zindi’s code review team. But more importantly, good code practice and good documentation are a critical skill for anyone working as a software or data engineer. Here are some tips for better code, from Zindi’s code review team.

**Reproducibility**

To ensure your model creates reproducible outputs, and reproduces the score you got on the leaderboard, you need to [set random seeds](https://towardsdatascience.com/how-to-use-random-seeds-effectively-54a4cd855a79) in your model. This is one step to ensure your code is reproducible.

If your submitted code does not reproduce your score on a Zindi leaderboard, we reserve the right to adjust your rank to the score generated by the code you submitted.

Another place where reproducibility could be impacted is in library package updates and in your environment, which we explore next.

**Libraries used**

Import all your libraries used at the top of your solution, whether that is in a script, or in one of the first cells in your notebook.

Import and initialise only the libraries used in the script. This reduces complexities and run time in your solutions.

Submit a [requirements.txt](https://www.freecodecamp.org/news/python-requirementstxt-explained/) file with your solution. A requirements file contains the version of the libraries you used. This is important especially if a library might be updated in the future. Again, only write the libraries you used in your solution.

**Coding environment**

It is important to identify where your code was run. This ensures that whoever is running your code, knows where to run it, especially if they are running your solution in a different environment and run into an error.

Examples of environments are your local computer (report your laptop specs), Colab, or Kaggle, to name a few.

**Data used**

Most competitions on Zindi limit you to the datasets found on the competition page. It is still important to note the datasets and where they came from. This is important in the event that external datasets were allowed, or in industry when you need to find the original datasets used.

**Data processing done in your code**

All data processing, no matter how simple, needs to be done in your solution. This ensures that your solution is reproducible, and that it can be put into production. Data should never be pre-processed in Excel or another programme.

**Accessibility to all**

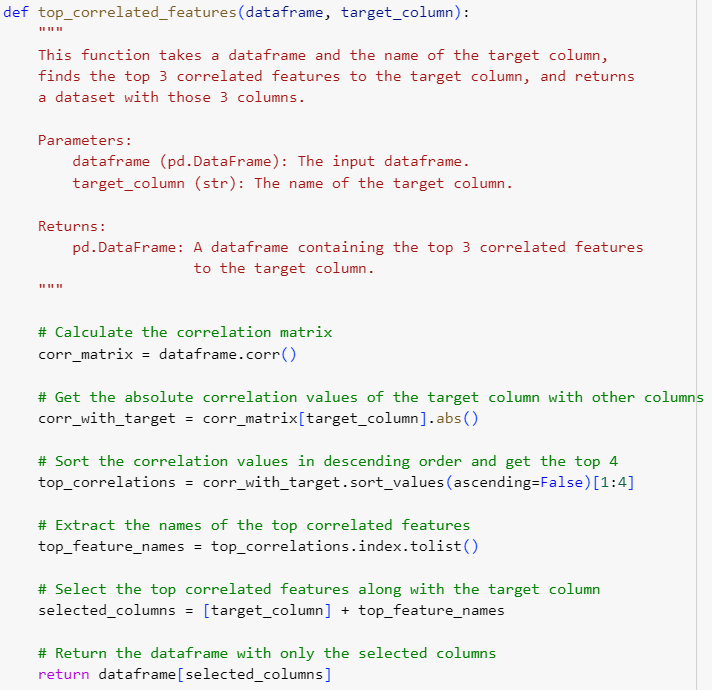
To ensure the Zindi environment is fair to all, regardless of resources available, you may only use freely available packages or products, unless otherwise specified in the competition.

**Coding practices**

If you are using a function, or a similar section of code more than once, create a method that does this. Name your method a suitable name.

If your function is complicated, consider [adding a mark-up section](https://stackoverflow.com/questions/2357230/what-is-the-proper-way-to-comment-functions-in-python) that explains what your function does. Good naming practices will help with this.

Here is an example of a well defined and commented method.



Features and variables need good names too. Even if you are creating a temporary variable, tmp is never an acceptable name.

Read this article on [best practices for writing code comments](https://stackoverflow.blog/2021/12/23/best-practices-for-writing-code-comments/).

**Data insights**

To secure your spot on the leaderboard you don’t need to submit your EDA, however clients find this information incredibly valuable.

At Zindi we recommend creating a script or notebook that shows some of your insights into the data and your interpretation. This step can go a long way in setting your solution apart in the client’s eyes.

**Final notes**

If your code does not run you will be dropped from the top 10. Please make sure your code runs before submitting your solution.

Writing good code is a continual learning experience, but we hope that these guidelines have been a helpful place to start. To keep improving, read up on articles and read other’s code to learn what you can do better.