Prize recipient solution documentation guide

Congratulations! You've gone up against dataheads from around the globe and emerged victorious! Laugh, dance, brush your shoulders off. You demonstrated serious skills, and helped make this world a better place in the process. Awesome job. Now you've finished in one of the top spots of the private leaderboard, which makes you eligible to receive a monetary prize. You're almost there.

In accordance with the official competition rules, the DrivenData terms of use, and applicable State and Federal law, we both have some due diligence to take care of before we can announce winners and disburse prizes.

You will receive a separate document to submit your legal documentation so that we can verify your legal identity. We will use this to verify your eligibility to participate and then review the specific laws and rules about giving out prizes based upon your nationality and our tax reporting obligations.

This document details the steps required to submit your solution materials.

I. **Code submission and result reproducibility**
   You package up and send us your code, documentation of dependencies, and any other assets you used. We review your package and make sure that it works and that we can fully reproduce the workflow from raw data to a submission comparable to your best submission. See the Competition prize winner submission template to get started.

II. **Basic information for winner announcement**
   Provide your preferred information for use in announcing the winners of the competition.

III. **Model documentation and write-up**
   You write up answers to our questionnaire, providing important context and documentation so that the beneficiary and the community get the most out of your work.

Please read this document carefully. Each section details exactly what is needed from you—the faster we can check all the boxes for our mutual responsibilities, the faster we can disburse your prize!

Thanks for your hard work, and congratulations for making it this far.

Best,
The DrivenData Team

I. Code submission and result reproducibility

You will need to submit a compressed archive of your code. You don't need to include the raw data we provided, but everything else should be included and organized. If the files are too large to be e-mailed, a Google Drive or Dropbox share (or other comparable method of transferring data) works.

Note: *please follow these instructions carefully*. The spirit and purpose of the competition (and the reason for offering prizes) is to give our beneficiary organizations the best possible solution *along with working code they can actually use*. In accordance with the competition rules, if we can't get your code working and reproduce your results with a reasonable effort, or if your entry is too disorganized to be practically usable, then your entry may be disqualified!

The overall concept is to **set up this archive as if it were a finished open source project**, with clear instructions, dependencies and requirements identified, and code structured logically with an obvious point of entry. Check out the competition prize winner README template to get started. We also have a data science project template which may be helpful.

At a minimum, your solution must include **an extremely clear README** that details all of the steps necessary to get to your submission from a fresh system with no dependencies (e.g., a brand new Linux, Mac OS X, or Windows installation depending on what environment you choose to develop under) and no other data aside from the raw data you downloaded from us.

This will probably entail the following:

- Necessary tools and requirements (e.g. "You must install Word2Vec 0.1c" or "Install the required Python packages in `requirements.txt`").
    - **All requirements should be clearly documented**, for instance in either a requirements.txt file with versions specified or environment.yml file.
- The series of commands, in order, that would get a reasonably experienced and savvy user from your code to a finished submission.
    - **Ideally, you will have a main point of entry to your code** such as an executable script that runs all steps of the pipeline in a deterministic fashion. A well-constructed Jupyter notebook or R script meets this standard.
    - **The next best thing is a list of specific, manual steps** outlining what to do. For example, "First, open Word2Vec and set these parameters. [...] Take the output file and run the script `src/make_preds.R` with the following parameters [...]". *(The limitations of this approach should be clear to any experienced data scientist!)*
- **Make sure to provide access to all trained model weights necessary to generate predictions from new data samples** without needing to retrain your model from scratch. Note that model weights can be contained in your archive or shared via a cloud storage service. The solution should provide clear instructions to perform inference on a new data point, whether or not it is included in the test set.
- Any other instructions necessary to end up with your winning submission file (or comparable — we understand that certain parts of model fitting are stochastic and won't result in exactly the same parameters every time).

II. Basic information for winner announcement

Please provide your preferred information for use in announcing the winners of the competition.

- Name (first and last name or first name and last initial): Lucas Robinet / Ziad Kheil
- Hometown: Toulouse, France.
- A recent picture of yourself or digital avatar (feel free to attach separately): find lrobinet.jpg, zkheil.jpg attached.
- Social handle or URL (optional): Lucas Robinet: https://www.linkedin.com/in/lucas-robinet-94ba45184/
- Ziad Kheil: https://www.linkedin.com/in/ziad-kheil-175279168/

III. Model documentation and write-up

Information included in this section may be shared publicly with challenge results. You can respond to these questions in an e-mail or as an attached file. Please number your responses.

1. Who are you (mini-bio) and what do you do professionally? If you are on a team, please complete this block for each member of the team.

   Lucas Robinet. *I am in my first year of doctoral studies in Toulouse at the CRCT and the IRT Saint-Exupéry. My topic is the multimodal analysis of cancer patients' data and more specifically of glioblastoma data using deep learning.*

   Ziad Kheil. *Currently a PhD student at the CRCT, I work on deep learning based medical image registration with a focus on thoracic images. Particularly in the case of 4D-CT phase registration, as well as multi-modal image registration.*

2. What motivated you to compete in this challenge?

   The multimodal aspect of the task, but also and above all the idea of being able to manipulate WSI and the resulting challenges.

3. High level summary of your approach: what did you do and why?

   We trained a ResNet to predict relapse, ulceration and breslow on low resolution images. Then concatenate the imagery embedding with a clinical embedding, obtained from dense layers. We used a focal loss to to handle class imbalance and help the model learn on difficult examples that are often present in WSI.

4. Do you have any useful charts, graphs, or visualizations from the process?

   No.

5. Copy and paste the 3 most impactful parts of your code and explain what each does and how it helped your model.

- The use of three losses, a focal loss on the relapse, a cross entropy on the breslow and on the ulceration to stabilize the training and obtain a latent space containing this information.
- The results are extremely tight, the study ablation conducted allowed us to improve the performance of our model by choosing the best hyper-parameters. As our model is quick to train and infer, this study could be as exhaustive as possible
- A logical phenomenon in the dataset was that older patients were more likely not to relapse within five years due to various factors. Since our model is based on standardized ages, it did not necessarily encode this information. Giving it to him directly allowed a slight improvement of the model.

6. Please provide the machine specs and time you used to run your model.
   - CPU (model): *AMD EPYC 7502P*
   - GPU (model or N/A): *NVIDIA GeForce 2080 Ti*
   - Memory (GB): *64 (cpu) / 11 (gpu)*
   - OS: *Ubuntu 20.04*
   - Train duration: *30 minutes*
   - Inference duration: *5 minutes*

7. Anything we should watch out for or be aware of in using your model (e.g. code quirks, memory requirements, numerical stability issues, etc.)?

   The seed for training our image feature extractor was not fixed. This has been corrected in the code template but it may explain the possible slight differences with the submitted scores.

8. Did you use any tools for data preparation or exploratory data analysis that aren't listed in your code submission?

   No.

9. How did you evaluate performance of the model other than the provided metric, if at all?

   Score AUC, precision, recall, f1-score, model calibration.

10. What are some other things you tried that didn't necessarily make it into the final workflow (quick overview)?

    Patch extraction, attention models, oversampling, cells segmentation, ensemble learning. Fractal and second-order patch analysis.

11. If you were to continue working on this problem for the next year, what methods or techniques might you try in order to build on your work so far? Are there other fields or features you felt would have been very helpful to have?

    Work at multiple scales of resolution. Unfortunately, we could not do it because of a too small storage space.

12. What simplifications could be made to run your solution faster without sacrificing significant accuracy?

    Our solution is already pretty quick

13. (optional) Whole Slide Images can be challenging to work with due to their size and the significant variation in their contents. What techniques and/or tools did you find helpful for working with WSIs and why?

    OTSU filtering and working with HSV images instead of RGB. HSV images allow to select only part containing cancerous cells. Analysis on fractal dimensions.