**Winners Guidelines for Cash Prize Competitions - Erick David Calderin Morales**

Congratulations on being in the top ranks! This document states the expectations for selecting winning models. These requirements are subject to revision for each competition and serve as a supplement to the general competition rules guidelines.

The documentation should be sent to the following email: [support@datasource.ai](mailto:support@datasource.ai)

We have 2 main components when choosing the winners:

1. **Model explanation**
2. **Model Submission**

**1- Model Explanation**

It is important to emphasize that your winning model will be studied by technical and non-technical people who are part of the company sponsoring the competition, so it should be informative for both audiences.

The model must be written in Python, and have readable code that follows the guidelines in point #2- Model Submission.

The final model explanation documentation must be in PDF or Word format. **It must be written in English** (unless another language is allowed) and must be well written and easily understandable. These are the questions you must answer and submit in the above format:

Background

1. Name: **Erick David Calderin Morales**
2. City/Country: **Córdoba/Argentina**
3. Email: **edcm.erick@gmail.com**
4. What studies have you done that are related to data science and machine learning?

**I have completed 5 courses about Data Science, Data Engineering and Deep Learning of 2 to 6 months of duration, the most recent was in Kaggle (ML30Days). Also this year I began to study a Data Science Master in a university.**

1. What relevant experience do you have in data science and/or machine learning?

**My most important experience consists of creation of small and medium projects for every course I have finished. There are other sources that add to my experience, for instance, Listening podcasts, reading Towards Data Science posts every day, watching Data Science videos on youtube and attending local and foreign meetings.**

1. Do you have experience related to the problem of this competition?

**No, I do not. It is my first competition, therefore It is my first project linked to predicting paper research scores.**

1. How much time have you invested in the competition?

**Although it started two months ago, I joined last month and since then I have been working all day.**

Model:

Answer the questions with 4-6 sentences summarizing the most important aspects of your model and analysis, such as:

PS: Please make external references to websites, blog posts and external resources where appropriate in your answers.

1. Provide a feature importance graph, showing the 10-20 most important features and partial graphs of the 3-5 most important features. If this is not possible, you should provide a list of the most important features.

| **Feature** | **Importance** | **Meaning** |
| --- | --- | --- |
| Year | **High** | Year published |
| en | **High** | Dummy  variables |
| es | **High** |
| pt | **High** |
| Text features | **Medium** | About 10.000 features |
| Pos features | **Medium** | About 28 features |

1. How did you select the features?

**The first four features of previous table were always included in input model because they contribute quietly to model prediction. In the next iterations I was trying how the model behaved if I added one more input or created combinations with them to reach the best prediction.**

1. Did you make any feature transformations that were important to the result?

**Yes, I did. It was necessary to apply One Hot Encoder method to Language because it is a categorical ordinal variable.**

**MinMax scaler was applied in Year column to transform the data into [0 - 1] range.**

1. Did you find any important interactions between the different features?

**Absolutely no. I did not find any correlation that helps me to reduce the error.**

**However, it was relevant that papers in English language represent almost 50% of the whole dataset.**

1. Did you use external data? (if allowed) **No, I did not.**
2. What tools and libraries did you use?

**I just used Spacy and NLTK which are not included in the Python core.**

1. What training method(s) did you use? (XGBoost, CatBoost, CNNs, RNNs)

**I used BayesianRidge model of Sklearn module. It was something unexpected because I was trusting that more advanced models like RandomForest, XGBoost generate a better prediction than the one I had in mind. And the most incredible thing here is that it was not necessary tuning hyperparameters.**

1. Did you ensemble the models? If yes, how did you balance the weights of the different models?

**No, I did not.**

1. Did you do stacking of models?

**No, I did not.**

1. What was the most important trick you used that gave the best result?

**A classic trick that I guess was used by other competitors consists in text vectorization. I concatenated the abstract and title to create a new column set by these two features.**

**But it crossed by my mind to create new features based on POS tags (Part Of Speech) of the text.**

**For instance: “This is my model” results “OBJ VERB NOM OBJ”.**

**I applied this procedure in each row storing the POS tags in a new column in order to vectorize later.**

**Here I created additional features such as number of sentences, stopwords and so on to reduce RMSLE value.**

**Finally, extracting crazy features took me a long time, about 40 minutes. But It was worth it.**

1. What do you think you did differently that set you apart in the competition?

**I think I did not struggle finding advanced or complicated models to train, nor used deep learning solutions to process the full text. I found out that the simplest model for this dataset was included in sklearn.linear\_model module.**

1. Did you make any important findings that are not mentioned in the previous questions?

**Yes, I did. I ran into that the abstract of papers with the highest score have the text in different languages. It is something weird and which could cause some rate of error in final prediction.**

1. Could a simplified model be made with your winning model? If so, is there a subset of features that can get 90-95% of your final performance? Which features would you select?

**Yes, it is possible. But there are no specific features to get a simplified model. I suggest increasing the quantity of “min\_df” hyperparameter in TfidfVectorizer to reduce both dimensionality and training time, but increasing the error just a little bit.**

1. How long does it take to train your model?

**The command %%time displays 2h 24m 16s hours of execution time.**

1. How long does your model take to make predictions?

**It takes about 1.54 seconds**

1. If you could make a simplified model, how long would it take to train it? And how long would it take to make a prediction?

**It would take between 40 and 60 minutes.**

**2- Model Submission**

Models should be submitted in a single .zip file containing all the items described below.

Below are some best practices for documenting and submitting your solution. There may be acceptable variations from these guidelines, depending on the type of competition, code or methods you have used. The main requirement is that you detail all the pieces necessary for the sponsor to reproduce your solution with the final score on the leaderboard within a reasonable margin.

This section is for a technical audience trying to replicate your solution. Please make sure your code is well commented and in English (variable names, methods, classes, comments, etc.).

**README.md**

Create this file at the top of the folder. Here is an example: <https://gist.github.com/danielmoralesp/75ba88ddfc6726bbfc5825706e77b1cc> This file should describe

1. The hardware you have used: CPU specifications, number of CPU cores, memory, GPU specifications, GPU number.
2. The operating system/platform used, including version number.
3. Any required third party software, including version numbers and installation steps. This can be provided as a Docker file rather than a section in the README.
4. How to train your model
5. How to make predictions on a new test set.
6. Important side effects of your code. For example, if your data processing code overwrites the original data.
7. Key assumptions of your code. For example, whether the output folder should be empty when starting a training run.

**Configuration file**

Create a subfolder with the necessary configuration files, such as `$HOME/.keras/keras.json`. The README should also include a description of what these files are and where they must be placed to work.

**requirements.txt**

Create a requirements.txt file at the top level of the archive. This is an example file: <https://gist.github.com/danielmoralesp/46b8cb6e9e8ad8e29c545e33dfeb108d>. This must specify the exact version of all packages used, such as `pandas==0.22.0`. This can be generated with tools like `pip freeze` in Python. The requirements file can also be replaced with a Dockerfile, as long as all installations use exact version numbers.

**directory\_structure.txt**

Create a directory tree reading at the top level of the file. Here is an example file: <https://gist.github.com/danielmoralesp/e933b33fa215ddc690d9a0f4b70c5ec2>.

**SETTINGS.json**

This file specifies the path to the training, test, model and output directories. This is an example file: <https://gist.github.com/danielmoralesp/12f129fa339915410861957fc49b9d45>.

* This is the only place that specifies the path to these directories.
* Any code that is doing I/O must use the appropriate base paths from SETTINGS.json.

**Copy of the serialized trained model**

Save a copy of the trained model to disk. This allows code to use the trained model to make predictions about new data points without having to retrain the model (which is usually much more time consuming). If the model checkpoints are part of your normal workflow, the REAMDE.md should indicate the path to the folder where you have saved them.

**entry\_points.md**

A list of the commands needed to execute your code. As a best practice, separate the training code from the prediction code. For example, using Python, there would be up to three entry points to your code:

1. Command: **python prepare\_data.py**
   1. which would
      1. Read training data from RAW\_DATA\_DIR (specified in SETTINGS.json)
      2. Run any preprocessing steps
      3. Save the cleaned data to CLEAN\_DATA\_DIR (specified in SETTINGS.json)
2. Command: **python train.py**
   1. which would
      1. Read training data from TRAIN\_DATA\_CLEAN\_PATH (specified in SETTINGS.json)
      2. Train your model. If checkpoint files are used, specify CHECKPOINT\_DIR in SETTINGS.json.
      3. Save your model to MODEL\_DIR (specified in SETTINGS.json)
3. Command: **python predict.py**
   1. which would
      1. Read test data from TEST\_DATA\_CLEAN\_PATH (specified in SETTINGS.json)
      2. Load your model from MODEL\_DIR (specified in SETTINGS.json)
      3. Use your model to make predictions on new samples
      4. Save your predictions to SUBMISSION\_DIR (specified in SETTINGS.json)