INFORMS 2020 QSR Data Challenge
CT Scan Diagnosis for COVID-19

Objective

The COVID-19 pandemic has changed lives around the world. Doctors are calling for the use of reliable and efficient methods to diagnose COVID-19 that would be faster and miss fewer cases than molecular testing of swab does. Computed tomography (CT) scans have been used for screening and diagnosing COVID-19. Chest CT scans have become a complementary candidate for the false negative issue from molecular testing. The goal of the 2020 INFORMS QSR Data Challenge is to diagnose COVID-19 using the chest CT scans. The Call for Participation can be found here.

Data Description

The data for this Data Challenge is selected from an open-source data set on COVID-19 CT images. The raw data have been divided into two subsets: training and test sets. The training dataset is provided to participants to develop their models. The training dataset consists of 251 COVID-19 and 292 non-COVID-19 CT images. In addition to the images, meta-information (e.g., patient information, severity, image caption) is provided in a spreadsheet. Participants can register for the challenge and the training data can be downloaded here. Participants can ONLY use the provided training dataset to develop their model. The test set is not provided to participants and will be used to evaluate the performance of the submitted models. The details of the original dataset can be found in Zhao et al. (2020).

Submission Instructions

Two websites will be provided, one for the execution of the model on the training data and the other for the submission of the final model. These websites will be available after July 15th, 2020. Participants will receive an email notification once sites are open. Each participating team is required to submit the model via the Data Challenge submission website. The submission must include:

1) Source codes in a Model.zip file that composes of Model.py or Model.R, Model.pickle/Model.h5 or Model.Rdata, Model.txt, and Requirements.txt with the detailed descriptions provided in the appendix;
2) A ReadMe.docx file that (i) explains how to run the model on the training data and (ii) reports the training performance in a confusion matrix;
3) A presentation of no more than 5 slides to briefly describe the method in the submitted model.

Evaluation

The participant models will be evaluated in two phases. The first phase evaluation verifies the model, and the second phase determines the finalists.

Phase 1: The submitted model will be evaluated on the provided training data. If the classification results (confusion matrix) generated by executing the submitted model agree with those reported in the submission, the participant is qualified for Phase 2. The time constraints on evaluating the training performance should be no more than 1 hour.

Phase 2: The model will be evaluated on the test data which is determined by the Organizing Committee. The time constraints for the classification of each image on the testing data should be no more than 10 seconds on average. The top 3 or 4 models will be selected as the finalists for the presentation at the 2020 INFORMS Annual Meeting.

Reference

Zhao, Jinyu, et al. "COVID-CT-Dataset: a CT scan dataset about COVID-19." arXiv preprint arXiv:2003.13865 (2020)

Appendix

We will utilize an automated platform for the evaluation of the submitted models. The evaluation platform supports two programming languages WITHOUT GPU: Python and R. The following files and descriptions are required:

a) Model.py or Model.R: this script is the entry function for your uploaded model. It is a single script with two functions: estimate(X_train, y_train), and predict(X_test, model). Here estimate(X_train, y_train) function will be used to estimate the proposed model, and predict(X, model) is to use a well-trained model for prediction. Here X_train and X_test are the lists (i.e., Python Lists or R Lists) of images in their original sizes, and y_train is a list of binary responses. All other customized modules (less than 10MB) in Python or dependencies in R can be uploaded with names and file structure specified in Model.txt description file.

- Example

```
# Example for Model.py
import numpy as np
import pandas as pd

# TODO: fillin the estimate function
def estimate(X_train,y_train):
        return model

# TODO: fillin the estimate function
def predict(X_test,model):
        return y_pred
```

b) Model.pickle/Model.h5 or Model.Rdata: this file should ONLY contain Python or R object of your trained model, which will be fed in predict(X_test, model) for prediction. For example: if you created a Sklearn model object “model”, please save a trained model as “Model.pickle” by using Pickle library in Python; If you created a deep learning model “model” in Tensorflow or Pytorch, please save the full trained model (i.e., not only save_weights) as “Model.h5”; If you created a model “model” in R, please save the trained model as “Model.Rdata”.

c) Model.txt: this text file contains the description for the proposed model, including details for execution and estimated time for training. This text file should exactly contain the following information:

```
# Model.txt
Customized module/dependency names in a list of relative
paths: [module1, module2, …]
Estimated time to run (in seconds):
```

- Example

```
# Example Model.txt
Customized module/dependency names in a list of relative
paths: [“utils/kronecker_product.py”,
“utils/inner_product.py”, “utils/my_reshape.py”]
Estimated time to run (in seconds): 256
```

d) Requirements.txt: this text file should list the dependencies with version specified which support the execution of your model script. Examples for Python and R are as follows:

- Python example

```
# Example requirements.txt for Python
python-dateutil==2.2
matplotlib==1.3.1
nose==1.3.0
numpy==1.8.0
pymongo==3.3.0
psutil>=2.0
```

- R example

```
# Example requirements.txt for R
ggplot2, version="0.9.1", repo="https://lib.ugent.be/CRAN/"
vioplot, version="0.3.4", repo="https://lib.ugent.be/CRAN/"
```