Making a difference in practice - Industrialization

Congrats! You have convinced higher management about the added value of your analysis/model. They would now like to practically use your model for customer selections in marketing campaigns. The next step in that process is to set your model in production! Explain what are the steps you think one would need to have the model you created in a production environment?

Putting machine learning models into production

Once the dataset exploration, the model and the research phase are completed comes the production phase. This phase requires to fully setup the data pipeline (the automatized path all needed data need to go through to be normalized and understandable by the model), the model training, deployment, and finally monitoring.

A key point is that the model has to be exportable to a format understandable by different platforms as the model is likely to be used in different environments. For example, **ONNX**, which stands for Open Neural Network Exchange, represents a model format for deep learning, using Google Protocol Buffers for the schema definition. Or more, **Pickle** is a python library which allows to export and import scikit-learn models using a process known as "pickling" to serialize a model hierarchy to a byte-stream (an viceversa).

The model has to be **trained** and the training strategy is extremely influenced by the dataset size and data availability (locally available, on the cloud, in real time, on clusters). Typically, batch training and real-time training (particularly useful when the model is already deployed and has to recalibrate when new data are collected) are the strategies most commonly adopted.

After the training phase the model is ready for predicting new data: this process is known as **model deployment**. Also in this stage new data can be predicted in real-time, as soon as they are collected, or in batches to predict chunks of data (when time is not a constraint). The scoring strategy depends on how the model has to be used. For example, considering the churn model developed in Part 1, an adequate strategy for both training and scoring (prediction) could be batch training and batch scoring and this if the model has to be used for predicting all customers churn likelihood. In this case, predicting if customers are likely to churn in real-time is an overkill as the prediction phase could run in batches over night and a bot or the marketing team could fetch all predictions the day after and perform the corresponding action to limit the customer likelihood to churn. However, if a model call has to be done whenever a customer calls the customers service, then a real-time strategy better suits.

For real-time predictions a common design pattern consists in building a **REST API**. Cloudera Data Science Workbench allows for wrapping the model and all its requirements into a

container which can be called using json requests and returns a predicted response.

Finally, after deployment into production, the model has to continuously be **monitored and updated** if efficiency drifts are recorded. Changes in the features distribution of new data are likely to produce wrong predictions as the model is not suited anymore for such patterns. Typically, model performance for new data is constantly monitored and is it falls below an acceptable threshold, then a new process to retrain the model is initiated, and that newly trained model is deployed.