David Müller **davmuell@yahoo.de**

Big Data Aspects of Random Forests

Random Forests are a bagging approach, i. e. they consist of many Decision Trees which are independent from another. In contrast to sequential models their training process could therefore be parallelized in a natural way. This means we could not only distribute the training on a single tree level but also amongst decision trees. Additionally, each tree in a Random Forest Regressor learns on a subsample of the data which could help to unnecessarily replicate the full dataset. Thus, Random Forests are indeed a good choice for a model that scales up or more precisely scales out. But sill, the dataset might be too large too read it into memory on a single machine. Even if the data could be read in, it might take forever to train the model, since RF often nee a lot of CPU power. In these cases the predictive model has to be built and trained on a cluster consisting of several nodes.

Before building the predictive model, the data must be stored and processed in an adequate manner. For big data the raw data will be probably stored in containers that maintain the data as objects (cloud object storage) taking advantage to use a cluster of machines. There are several popular options for example AWS S3 or Azure Blob Storage or Google Cloud Storage. One drawback is obviously that these cloud services have to be paid and can be costly.
For processing and analyzing big data Apache Spark is a good framework. It is multi-language based, e. g. SQL, Python, Scala, R, . . . can be used. Databricks provides a nice cloud-based environment for running Spark in combination with Python. However, in order to create and use computational efficient and powerful clusters, we would have to pay for the service. The integrated Databricks File System (DBFS) enables to interact with files stored in cloud object storage. Once the data is in the DBFS it can be stored in delta format. Based on that format we could work with delta tables which acts like traditional relational databases, since Delta Lake framework adds transaction support to Data Lakes. Basically, delta format is a column-oriented format (Parquet) plus delta-transformation-logs. Column-orientation yields usually to lower file sizes and faster query time resulting in significantly reduced cloud costs. Therefore, by relying on Databricks together with for example Azure and Spark (Pyspark), the feature preprocessing and selection could be done in similar steps and syntax as in Part 1. Moreover, Apache Spark offers a scalable Machine Learning library (MLlib), where Random Forests Regressors are integrated.

In conclusion, the steps for the scaled-up solution would be to upload the data into the DBFS, then to save it in delta format. After that do the feature processing and selection steps as in Part 1, use the MLlib to build the Random Forest Regressor and train the model. Limitations and drawbacks arise from additional costs for cloud storage and running the computationally efficient clusters. Note that there is no easy way to train Random Forests online, compared to e. g. Bayesian Regression. Hence, we would use the new daily observations only for predictions but not for learning, if not we retrain the complete model.
I have some hands-on experience in using Databricks and Apache Spark (Pyspark) from two projects, and a solid understanding of the workflows. However, I would not describe myself as an expert with these technologies yet as I've only worked with them for several months.