Phase 3 - deployment

This notebook will provide and overview how to deploy and predict the CPE in two ways

- The model was build/export in the last notebook (Phase_2_Advanced_Analytics__predictions)
  This notebook show another option to save/export the model using the H2O flow UI and complement the information with deployment for predictions.

The predictions will be presented in 2 ways

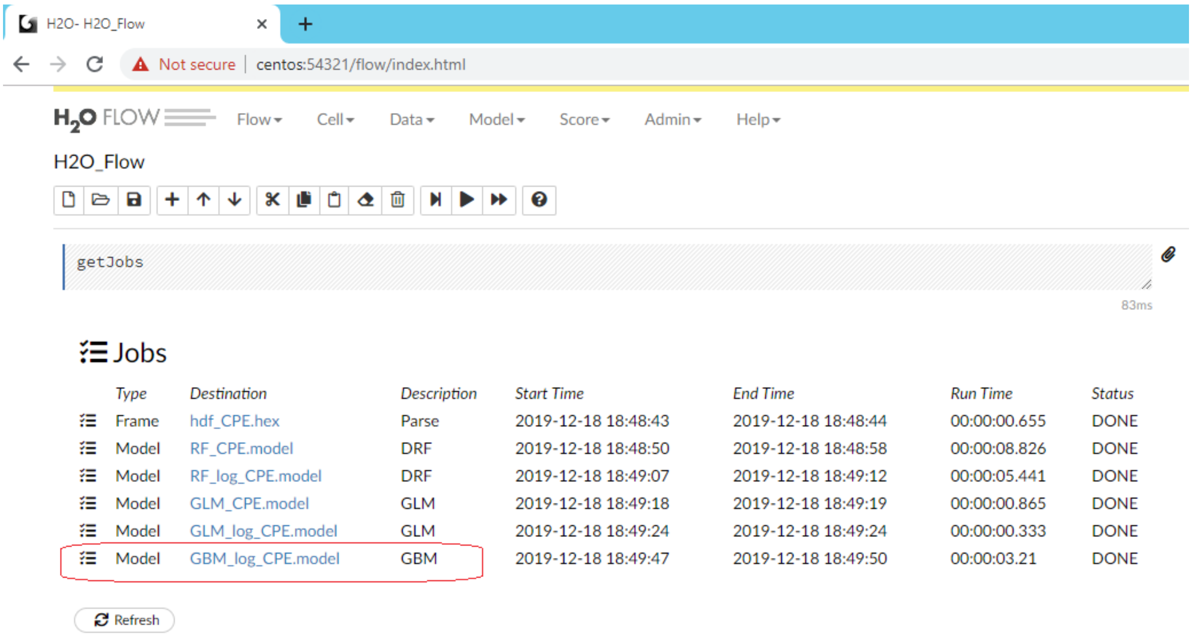
- Batch process
- Online / real time predictions

Export model: Export the model GBM (best performance) using H2O flow UI as detailed below

In [2]:

```
from IPython.display import Image
Image(filename='./data/H2O-FLOW-UI-GBM-MODEL.PNG')
```

Out[2]:



```
getJobs
```

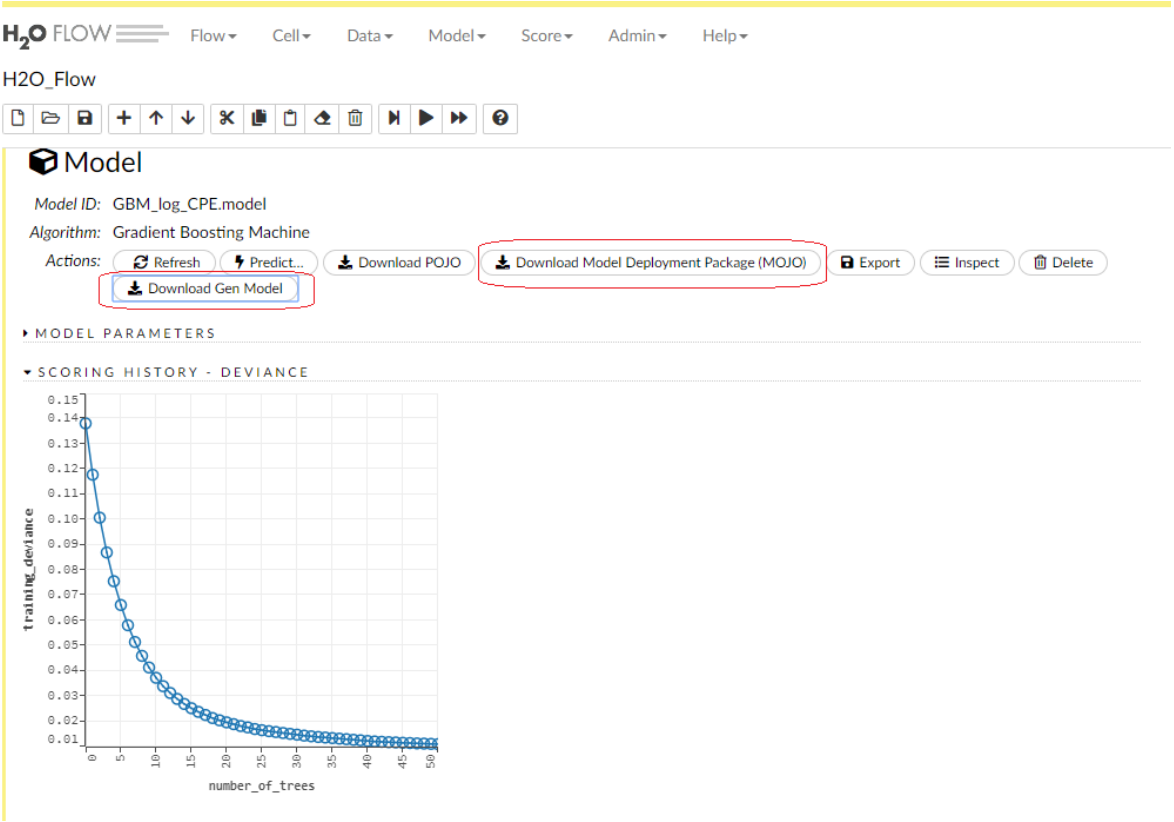
Jobs

	Type	Destination	Description	Start Time	End Time	Run Time	Status
☰	Frame	hdf_CPE.hex	Parse	2019-12-18 18:48:43	2019-12-18 18:48:44	00:00:00.655	DONE
☰	Model	RF_CPE.model	DRF	2019-12-18 18:48:50	2019-12-18 18:48:58	00:00:08.826	DONE
☰	Model	RF_log_CPE.model	DRF	2019-12-18 18:49:07	2019-12-18 18:49:12	00:00:05.441	DONE
☰	Model	GLM_CPE.model	GLM	2019-12-18 18:49:18	2019-12-18 18:49:19	00:00:00.865	DONE
☰	Model	GLM_log_CPE.model	GLM	2019-12-18 18:49:24	2019-12-18 18:49:24	00:00:00.333	DONE
☰	Model	GBM_log_CPE.model	GBM	2019-12-18 18:49:47	2019-12-18 18:49:50	00:00:03.21	DONE

In [3]:

```
from IPython.display import Image
Image(filename='./data/H2O-FLOW-UI-GBM-MODEL-download.PNG')
```

Out[3]:



Model ID: GBM_log_CPE.model

Algorithm: Gradient Boosting Machine

Sample of new campaigns to be predicted

In [21]:

```
import pandas as pd
df = pd.read_csv('./GBM_MODEL/New_campaings_for_predictions.csv')
df.tail(10)
```

Out[21]:

	LineItemsID	URL	xyz_campaign_id	channel	channel_ad_id	gender	age	interest	spen
884	885	URL-1	1178	5	14	F	25-34	Interest - 104	11
885	886	URL-1	1178	4	11	F	25-34	Interest - 105	23
886	887	URL-1	1178	11	1	F	25-34	Interest - 106	2
887	888	URL-1	1178	4	13	F	25-34	Interest - 107	16
888	889	URL-1	1178	1	4	F	25-34	Interest - 108	3
889	890	URL-1	1178	2	13	F	25-34	Interest - 109	25
890	891	URL-1	1178	11	3	F	25-34	Interest - 110	12
891	892	URL-1	1178	11	2	F	25-34	Interest - 111	2
892	893	URL-1	1178	2	8	F	25-34	Interest - 113	13
893	894	URL-1	1178	2	5	F	25-34	Interest - 114	11

Important attention point

- All information will be provided for prediction (base information available in the simulated/demo data) however just the relevant information were used during the model build detailed in the Notebook: Phase_2_Advanced_Analytics__predictions
- For example LineItemsID is just an index number and do not provide relevant information and is not going to be used for prediction

Batch Prediction: Generate prediction for new data

To execute the prediction as presented below it is not necessary to have an H2O cluster running

The processo show below was executed in 2 steps to show in detail the process but in production environment this process must be executed in just one step

Simulation in 2 steps

Step 1. batch process to run the java program

Step 2. python program to link the new data and the predictions with the CPE

Can be used any programming language to run the prediction and get the results (such as R, Python, Java, C#, ...)

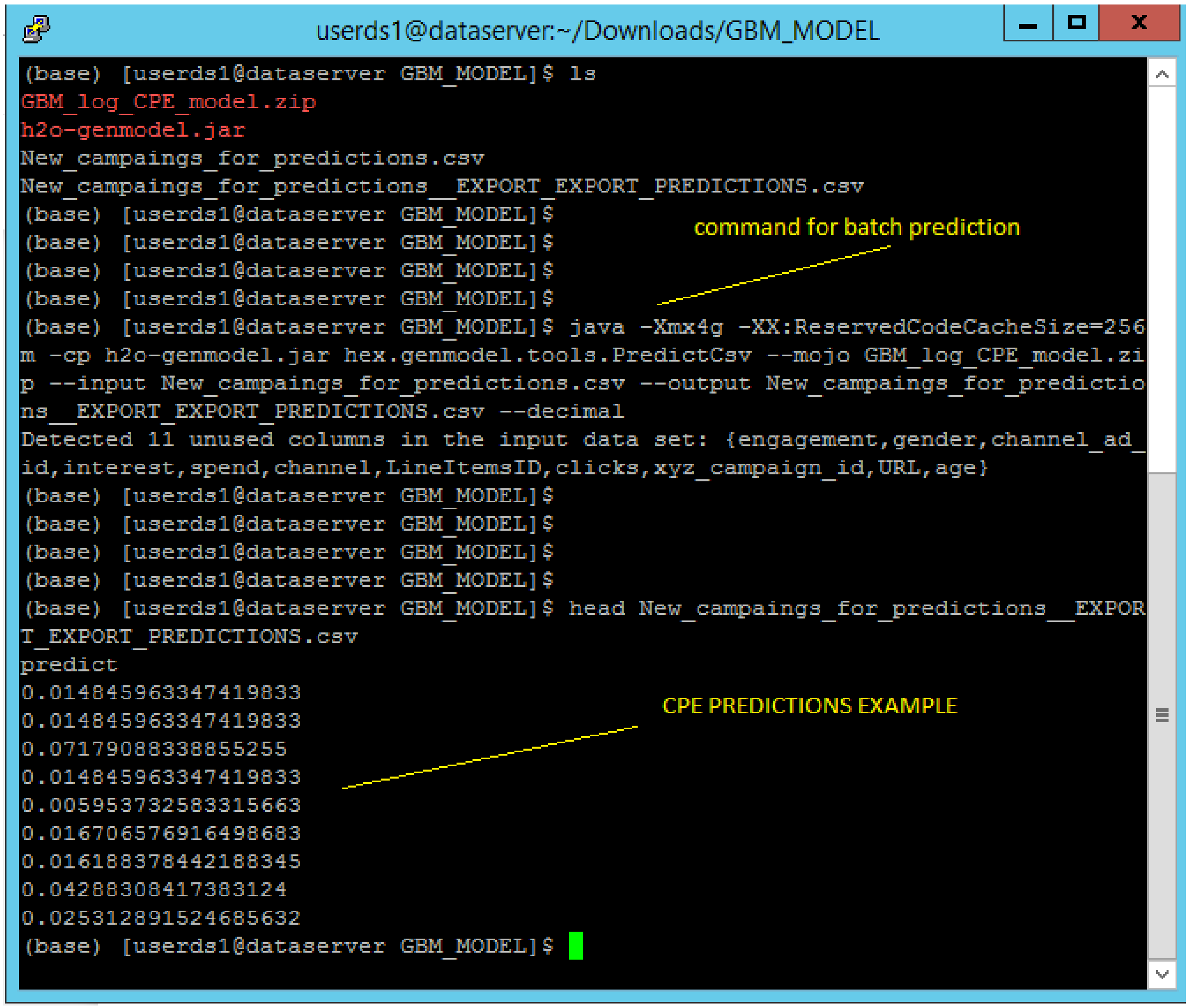
Run batch java process to gererate/score the predictions of CPE

```
To generate prediction (CPE) for new data just run the command

## EXAMPLE
java -Xmx4g -XX:ReservedCodeCacheSize=256m -cp <h2o-genmodel.jar_EXPORTED_ABOVE>
hex.genmodel.tools.PredictCsv --mojo <GBM_log_CPE_model.zip_EXPORTED_ABOVE> --input
INPUT_FILE_FOR_PREDICTION.csv --output
OUTUPUT_FILE_WITH_PREDICTIONS_FOR_CPE__EXPORT_EXPORT_PREDICTIONS.csv --decimal

## REAL PREDICTION
java -Xmx4g -XX:ReservedCodeCacheSize=256m -cp h2o-genmodel.jar
hex.genmodel.tools.PredictCsv --mojo GBM_log_CPE_model.zip --input
New_campaings_for_predictions.csv --output
New_campaings_for_predictions__EXPORT_EXPORT_PREDICTIONS.csv --decimal
```

In [7]:

```python
from IPython.display import Image
Image(filename='./data/Batch-prediction-h2o.PNG')
```

Out[7]:



```
(base) [userds1@dataserver GBM_MODEL]$ ls
GBM_log_CPE_model.zip
h2o-genmodel.jar
New_campaings_for_predictions.csv
New_campaings_for_predictions__EXPORT_EXPORT_PREDICTIONS.csv
(base) [userds1@dataserver GBM_MODEL]$
(base) [userds1@dataserver GBM_MODEL]$
(base) [userds1@dataserver GBM_MODEL]$
(base) [userds1@dataserver GBM_MODEL]$
(base) [userds1@dataserver GBM_MODEL]$ java -Xmx4g -XX:ReservedCodeCacheSize=256
m -cp h2o-genmodel.jar hex.genmodel.tools.PredictCsv --mojo GBM_log_CPE_model.zi
p --input New_campaings_for_predictions.csv --output New_campaings_for_predictio
ns__EXPORT_EXPORT_PREDICTIONS.csv --decimal
Detected 11 unused columns in the input data set: {engagement,gender,channel_ad_
id,interest,spend,channel,LineItemsID,clicks,xyz_campaign_id,URL,age}
(base) [userds1@dataserver GBM_MODEL]$
(base) [userds1@dataserver GBM_MODEL]$
(base) [userds1@dataserver GBM_MODEL]$
(base) [userds1@dataserver GBM_MODEL]$
(base) [userds1@dataserver GBM_MODEL]$ head New_campaings_for_predictions__EXPOR
T_EXPORT_PREDICTIONS.csv
predict
0.014845963347419833
0.014845963347419833
0.07179088338855255
0.014845963347419833
0.005953732583315663
0.016706576916498683
0.016188378442188345
0.04288308417383124
0.025312891524685632
(base) [userds1@dataserver GBM_MODEL]$
```

command for batch prediction

CPE PREDICTIONS EXAMPLE

Sincronize all information - new campaign data and new predictions for CPE

- Remember that the prediction was done in logarithmic scale and now is necessary to rever the result with exponential function

In [22]:

```
CPE_predictions = pd.read_csv('./GBM_MODEL/New_campaings_for_predictions__EXPORT_EXPORT_PRE
CPE_predictions.tail()
```

Out[22]:

	predict
889	1.324483
890	0.783198
891	0.459563
892	0.952897
893	0.794780

In [25]:

```
import numpy as np
df['CPE_predition_LOG'] =  CPE_predictions['predict']
df['CPE_predition'] = round(np.exp(CPE_predictions['predict']) -1, 3)
df.tail()
```

Out[25]:

	LineItemsID	URL	xyz_campaign_id	channel	channel_ad_id	gender	age	interest	spen
889	890	URL-1	1178	2	13	F	25-34	Interest - 109	25
890	891	URL-1	1178	11	3	F	25-34	Interest - 110	12
891	892	URL-1	1178	11	2	F	25-34	Interest - 111	2
892	893	URL-1	1178	2	8	F	25-34	Interest - 113	13
893	894	URL-1	1178	2	5	F	25-34	Interest - 114	11

Online prediction: Generate prediction for new data

The online prediction could be implemented using diferent architectures such as

1. Serverless function such as Amazon AWS Lambda + API Gateway https://aws.amazon.com/lambda/?nc2=h_ql_prod_fs_lbd (https://aws.amazon.com/lambda/?nc2=h_ql_prod_fs_lbd)

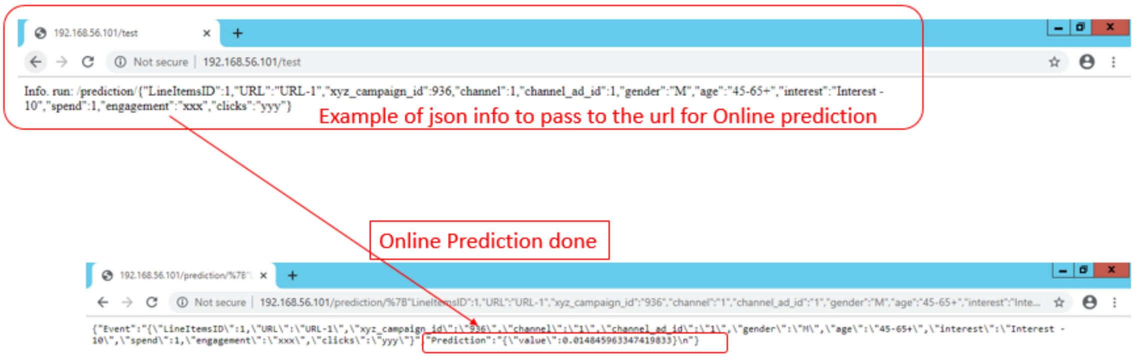
2. Java program that use POJO/MOJO model for online prediction
   http://docs.h2o.ai/h2o/latest-stable/h2o-docs/productionizing.html#step-2-compile-and-run-the-mojo (http://docs.h2o.ai/h2o/latest-stable/h2o-docs/productionizing.html#step-2-compile-and-run-the-mojo)
3. Microservices architecture using Docker (python + flask app + NGINX for load balance)
   Could be implemented on-premise solution or even using cloud solutions such as container orchestration as GKE (Google Kubernetes Engine)
   https://cloud.google.com/kubernetes-engine/ (https://cloud.google.com/kubernetes-engine/)

The solution presented below show the prediction done trought one json information passed to the URL
This API could be deployed in any of the 3 options detailed above

In [1]:

```
from IPython.display import Image
Image(filename='./data/Online-Prediction.PNG')
```

Out[1]:



192.168.56.101/test
Not secure | 192.168.56.101/test

Info. run: /prediction/{"LineItemsID":1,"URL":"URL-1","xyz_campaign_id":936,"channel":1,"channel_ad_id":1,"gender":"M","age":"45-65+","interest":"Interest - 10","spend":1,"engagement":"xxx","clicks":"yyy"}

Example of json info to pass to the url for Online prediction

Online Prediction done

192.168.56.101/prediction/%7B"
Not secure | 192.168.56.101/prediction/%7B"LineItemsID":1,"URL":"URL-1","xyz_campaign_id":"936","channel":"1","channel_ad_id":"1","gender":"M","age":"45-65+","interest":"Inte...

```
{"Event":"{\"LineItemsID\":1,\"URL\":\"URL-1\",\"xyz_campaign_id\":\"936\",\"channel\":\"1\",\"channel_ad_id\":\"1\",\"gender\":\"M\",\"age\":\"45-65+\",\"interest\":\"Interest -
10\",\"spend\":1,\"engagement\":\"xxx\",\"clicks\":\"yyy\"}","Prediction":"{\"value\":0.014845963347419833}\n"}
```

Summary and final considerations

The model build in Phase 2 and also exported in this notebook can be deployed for batch and online predictions

- Batch process => the batch process is the way to go to predict large ammount of campaigns and for back-office analysis using some BI tools
- Online prediction => The online prediction using microservices architecture for example, is the way to go if the company has online interfaces integrated with lauch campaign programs. With this approach is possible to analyse specific campaign prediction

In [ ]:

```

```