



Machine Learning for Physicists

Hands-on session

Day 1

Moscow International School of Physics 2024

Vladimir Bocharnikov, Alexey Boldyrev,
Evgeniy Kurbatov, Fedor Ratnikov

Our Team



LHCb, SHiP,
NICA, c-tau

Previously:
ARGUS, SSC,
HERA-B, CDF, CMS

Fedor Ratnikov



BM@N

Previously:
COMET, ILD, CALICE

Vladimir Bocharnikov



LHCb

Previously:
ATLAS

Alexey Boldyrev



SHiP, LHCb

Previously:
CLAS12

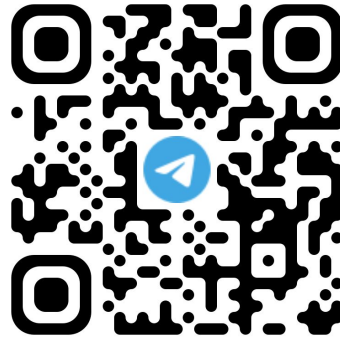
Evgeniy Kurbatov

Hands-on session

Day 1 - General Data Analysis

Practice of Data Analysis using Yandex DataSphere:
<https://datasphere.yandex.ru>

Connection instructions and announcements are in:



[ML@MosPhys](#) Telegram chat



Hands-on session: JupyterLab @DataSphere

LAMBDA • HSE

The screenshot shows a JupyterLab interface. At the top, there's a navigation bar with 'HSE Lambda school', 'Participant ajet2o4nfdk00lsje5n', and 'JupyterLab'. Below that is a menu bar with 'File', 'Edit', 'View', 'Run', 'Kernel', 'Git', 'Snippets', 'Tabs', 'Settings', and 'Help'. The main area is divided into a file browser on the left and a notebook editor on the right. The file browser shows a directory structure with files like 'scheme.png', 'Seminar1_intro.ipynb', and 'untitled.txt'. The notebook editor shows a notebook titled 'Seminar1_intro.ipynb' with the following content:

Seminar 1. Introduction

Jupyter notebooks

This is an IPython notebook, running a python kernel underneath.

The state of the python interpreter, all the variables, class and function definitions are preserved between the cell executions. In case you want to reset that state (e.g. to check whether your code works correctly when the cells are executed in the direct order), go to `Kernel -> Restart kernel...`

The most useful features of Jupyter:

- contextual help (hit `TAB`)
- quick access to documentation (`SHIFT+TAB`)

Numpy and vectorized computing

Almost any machine learning model requires some computational heavy lifting usually involving linear algebra problems. Unfortunately, raw python is terrible at this because each operation is interpreted at runtime.

So instead, we'll use `numpy` - a library that lets you run blazing fast computation with vectors, matrices and other tensors. The god object here is `numpy.ndarray`:

please keep running all the code cells as you read

```
[1]: import numpy as np

a = np.array([1,2,3,4,5])
b = np.array([5,4,3,2,1])
print("a = ", a)
print("b = ", b)

# math and boolean operations can be applied to each element of an array
print("a + 1 =", a + 1)
print("a * 2 =", a * 2)
print("a == 2", a == 2)
# ... or corresponding elements of two (or more) arrays
print("a + b =", a + b)
print("a * b =", a * b)
```