Declaration of Authorship

I hereby declare that the thesis submitted is my own unaided work. All direct or indirect sources used are acknowledged as references. This paper was not previously presented to another examination board and has not been published.

I am aware that the thesis in digital form can be examined for the use of unauthorized aid and in order to determine whether the thesis as a whole or parts incorporated in it may be deemed as plagiarism. For the comparison of my work with existing sources I agree that it shall be entered in a database where it shall also remain after examination, to enable comparison with future theses submitted.

Meschede, 10th September 2023.

Casimir Giesler
MatNr: 123454678
Email: curie.marie@fh-swf.de
Corresponding Author

This Declaration must be signed by all listed Authors to be valid.

Math stuff for pyspark

1 Simulation of a Dataset

In order to generate a large dataset which fulfills the requirements ($n \gg 10^9, k \gg 10^5$), the generation of the values needs to be done in a distributed fashion. At first, the following values need to be initialized:

- `n` - number of rows/samples
- `k` - number of colums/features
- $\vec{\beta}$ - beta, the coefficients of the function
- `cov` - a covariance vector that determines the covariance to the first column for each column

In this implementation, `n` and `k` need to be set by the user while $\vec{\beta}$ and `cov` are generated by numpy. For generating the actual dataset, `pyspark.mllib.random.RandomRDDs.normalVectorRDD(sc, n, k)` is used. This function creates an `rdd` containing `n` vectors, each containing `k` entries, where each entry is generated from a standard-normal distribution.

After generating this random noise matrix, the user-defined-function `createRow(noise)` is applied to the `rdd`, which returns two values, $\vec{x}$ (1) and y (2).

With `noise` as ϵ and `cov` as c:

$$\vec{x} = (\epsilon_0, \epsilon_0 c_1 + \epsilon_1, \ldots, \epsilon_0 c_i + \epsilon_i) \tag{1}$$

$$y = \vec{x} \cdot \vec{\beta} \tag{2}$$

Applying this function produces an RDD where the first element is the x-vector, and the second element is the target variable. The resulting feature matrix (consisting out of n $\vec{x}$ vectors) therefore consists out of k columns, where every column is linearly dependent on the first column, plus additional noise. An exemplary distribution is visualized in figure 1.

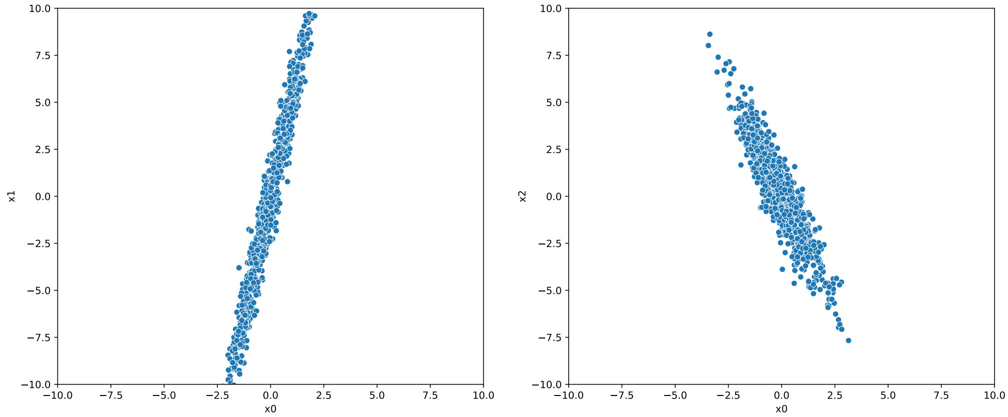

Figure 1: *exemplary generated dataset*