# The code to deploy the model as an API

## Deploy the model as an API on your own computer (command prompt)

1. Execute the JupiterNotebook « Implementez\_modele\_scoring ».
2. Open the command prompt window (I use Anaconda Command prompt, in other command prompts the deployment can work differently)
3. Go to the folder with your project (in this case P7\_Berezhnaia\_Veronika). Use the command cd Folder\_name/Another\_folder\_name/P7\_Berezhnaia\_Veronika
4. Type the command: mlflow models serve -m mlflow\_model/
5. To check that everything works correctly: you receive a response telling where your model serves. It includes a host, a port and an identifier.
6. Open the new command prompt window.
7. You don’t have to go to a specific folder right now. Send a curl request. An example of a request is in API\_command\_prompt\_json\_request.txt file. In case you don’t have this file, use the request example from below, but make sure you don’t have “new line” symbols. Your whole request should be on one line.

curl http://127.0.0.1:5000/invocations -H "Content-Type: application/json" -d "{\"columns\": [\"NAME\_INCOME\_TYPE\", \"AMT\_ANNUITY\", \"EXT\_SOURCE\_1\", \"EXT\_SOURCE\_2\", \"EXT\_SOURCE\_3\", \"DAYS\_EMPLOYED\_PERC\", \"CREDIT\_TERM\", \"STATUS\_0\_SUM\", \"STATUS\_1\_SUM\", \"STATUS\_2\_SUM\", \"STATUS\_3\_SUM\", \"STATUS\_4\_SUM\", \"STATUS\_5\_SUM\", \"STATUS\_C\_SUM\", \"STATUS\_X\_SUM\", \"CREDIT\_ACTIVE\_Active\_SUM\", \"CREDIT\_ACTIVE\_Bad\_debt\_SUM\", \"CREDIT\_ACTIVE\_Closed\_SUM\", \"CREDIT\_ACTIVE\_Sold\_SUM\", \"CREDIT\_TYPE\_Another\_type\_of\_loan\_SUM\", \"CREDIT\_TYPE\_Cash\_loan\_(non-earmarked)\_SUM\", \"CREDIT\_TYPE\_Consumer\_credit\_SUM\", \"CREDIT\_TYPE\_Credit\_card\_SUM\", \"CREDIT\_TYPE\_Microloan\_SUM\", \"CREDIT\_TYPE\_Unknown\_type\_of\_loan\_SUM\", \"DAYS\_BIRTH\", \"FLAG\_EMP\_PHONE\", \"FLAG\_WORK\_PHONE\", \"FLAG\_PHONE\", \"REGION\_RATING\_CLIENT\_W\_CITY\", \"REG\_CITY\_NOT\_LIVE\_CITY\", \"REG\_CITY\_NOT\_WORK\_CITY\", \"LIVE\_CITY\_NOT\_WORK\_CITY\", \"FLAG\_DOCUMENT\_2\", \"FLAG\_DOCUMENT\_3\", \"FLAG\_DOCUMENT\_6\", \"FLAG\_DOCUMENT\_9\", \"FLAG\_DOCUMENT\_13\", \"FLAG\_DOCUMENT\_14\", \"FLAG\_DOCUMENT\_16\", \"FLAG\_DOCUMENT\_21\", \"F\_AGE\", \"M\_AGE\", \"CNT\_ADULTS\"], \"data\":[[\"Working\", 10, 0.1, 0.1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1]]}"

1. To check that everything works correctly: you receive the response which is a list with a positive prediction [1.0], difficulties to pay, or with a negative prediction [0.0], all other cases.
2. You can send as many requests as you want.

## Dashboard and its interactions with the API (both on your computer)

Open the dashboard on your own computer, send the request from the dashboard to the API on your own computer, receive the answer to the dashboard.

Follow the points 1-6 from the section “Deploy the model as an API on your own computer (command prompt)”.

1. Go to the folder with your dashboard (in this case P7\_Berezhnaia\_Veronika). Use the command cd Folder\_name/Another\_folder\_name/P7\_Berezhnaia\_Veronika
2. Type the following command: streamlit run dashboard.py
3. To check if everything is correct: the dashboard will open in a new tab in your default browser.
4. To interact with the API, click the “Predict” button on the dashboard. You receive the response which is a list with a positive prediction [1.0], difficulties to pay, or with a negative prediction [0.0], all other cases.

## Deploy the model as an API on your own computer (Jupyter Notebook)

1. In a Jupyter Notebook cell, when you type exclamation mark !, it transfers the following content to Anaconda command prompt.
2. Type the following command: ! mlflow models serve -m mlflow\_model/
3. The execution of the cell will be suspended (you will see the asterisk in the cell)
4. Write in the next cell the exclamation mark ! followed by a space and the curl request (see the example above in the section “Deploy the model as an API on your own computer (command prompt)”
5. Interrupt the kernel. This is necessary, if you execute the cell with the curl request after the model serve cell, it we wait in the queue. As the model serve cell is suspended, the curl request cell will be suspended as well.
6. Execute the cell with the curl request.
7. You can send as many requests as you want. If it doesn’t work, reexecute the previous cell with the model serve command.