Group project guideline

The project is carried out in a group of no more than 3 students.

To perform laboratory work, you need a subscription to a basic or standard level of Microsoft Azure ML, a computing device with Internet access, and files with source data.

Azure ML allows you to create experiments in which you can manage data, create machine learning models and visualize results. In this lab, we will develop an end to end solution with cloud backend. This lab is suitable for half day training depending on the level of users' ML knowledge. Based on an OpenSource dataset, we will develop a digit recognition Azure ML solution, publish it as web service, connect with Azure Management API to manage the usage rate, overcome CORS and security issues, and integrate into a simple Azure Web application to draw our own characters on a web page canvas and retrieve the ML prediction about the character. In this lab we will use the MINST, publicly available large database of handwritten digits, to develop our ML model.

We will also give an overview about feature engineering based on this case study.

This lab aims to demonstrate how to develop a simple end to end solution that uses Azure ML solution. Having different type of datasets, we will focus on image understanding and explore the possible feature extraction process.

Report on the implementation of laboratory work

As a report on laboratory work, you must provide screenshots of the stages of laboratory work. **The report must include 10 screenshots, which are marked in the laboratory guideline with a red border.**

Report in a file format .docx must be uploaded to the LMS – Group project mod 4. Before uploading, specify your last name (without spaces or other characters) as the file name.

Activity on seminars assessment policy

The lab report must be uploaded before the end of the class according to the schedule.

In this case, if **there are 10 screenshots confirming the fact of laboratory work completion, the student receives a score of 10 (1 point for each screenshot).**

When uploading lab work, the exact upload time is recorded in the LMS. Those who did not complete the work during the classes are given the opportunity to complete the work before the end of the day 29.04.2020 23:59:59.

Works uploaded later than the end of the day will not be checked.

Login to Azure ML Studio

1. Follow the link https://studio.azureml.net and log in under your Microsoft account associated with your Azure ML subscription (enter your ****@edu.hse.ru e-mail address as a login and the password)
2. If the welcome page appears on the screen, close it by pressing OK button. If then a page containing a collection of sample experiments (Microsoft Samples) appears, close it too, by pressing the "×" button.
3. After that, your screen should look like the one shown in fig. 1.

Experiment creation

1. Press the "+ NEW" button, which is located in the lower left corner of the Azure ML Studio environment. Then select the Experiment category and from the collection of sample experiments (Microsoft Samples)

select an empty experiment - “Blank Experiment”, which looks like shown in fig. 2.

2. Change the name of the experiment - instead of the default name “Experiment created on today’s date” write “ODR 01 Read One Digit Raw Data”.

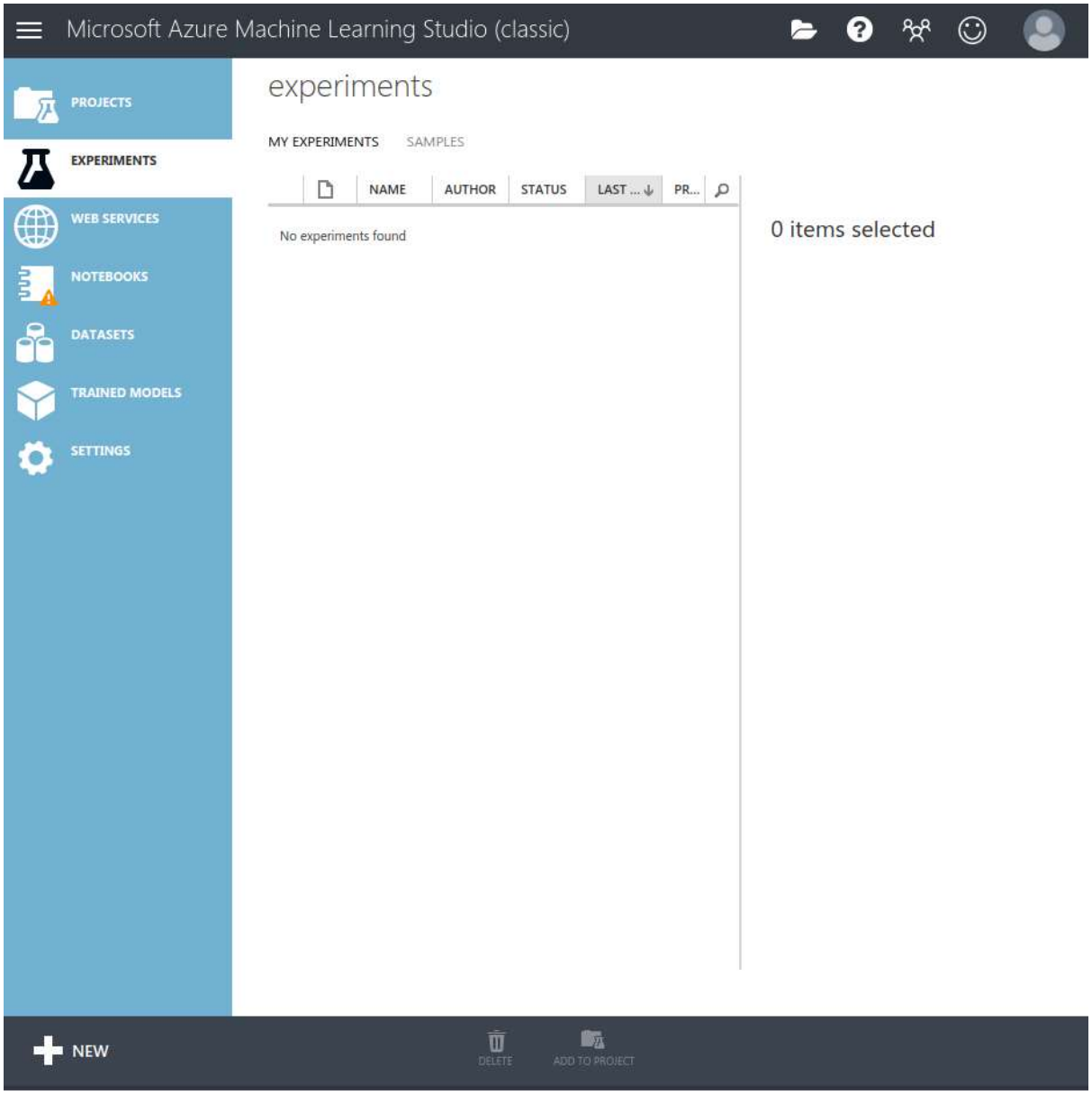


Fig. 1 Experiment creation

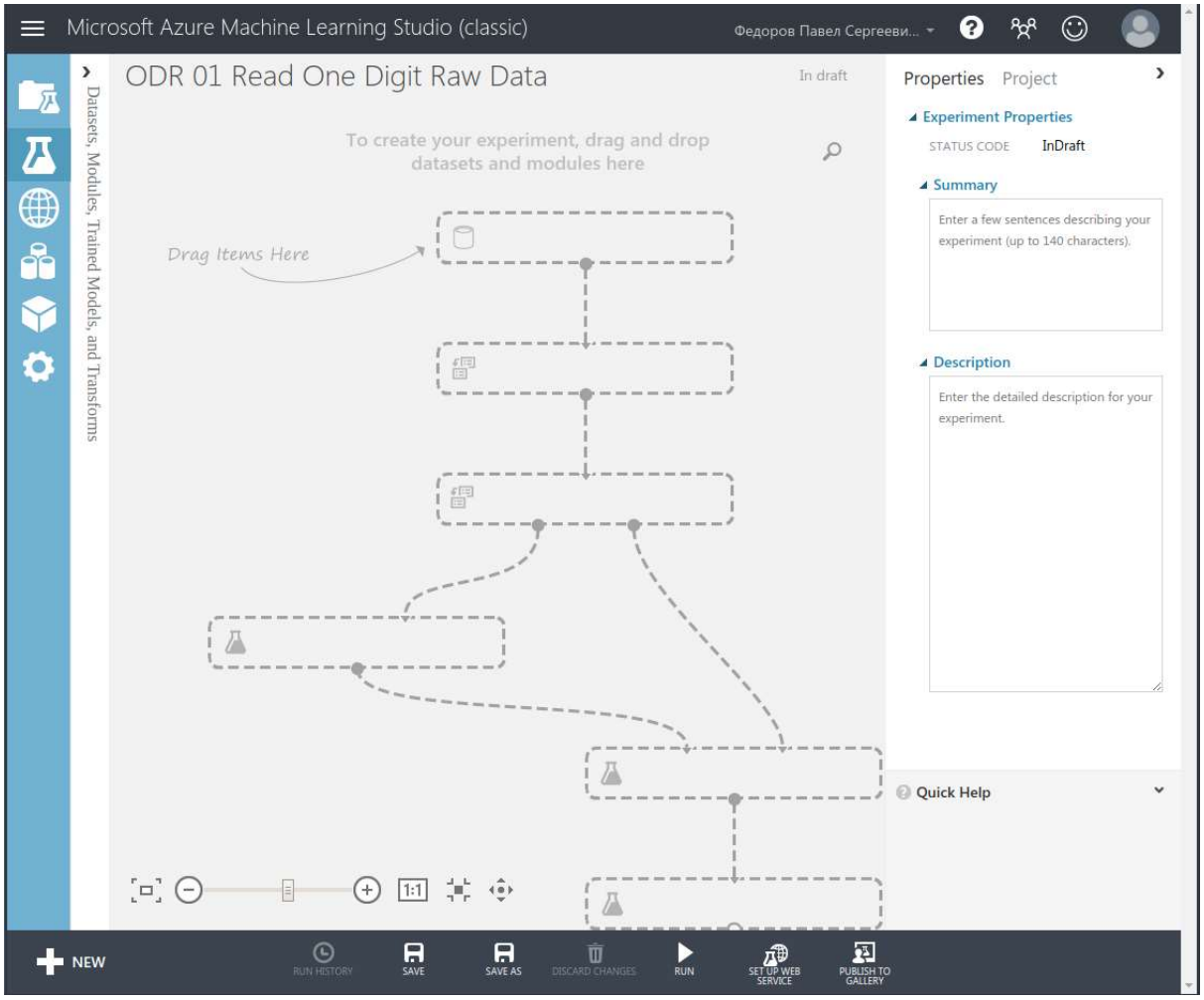
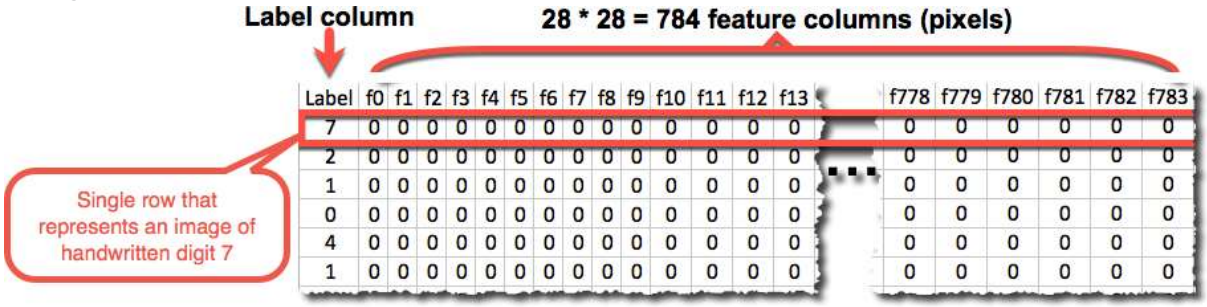


Fig. 2

Exploring and Understanding the Dataset

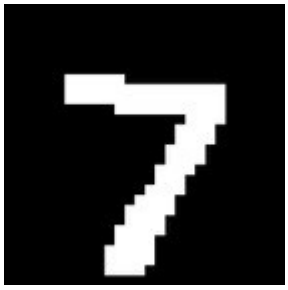
MINST handwritten digit images database is also available, ready to use, in Azure ML workspace as one the other sample datasets. Database consists of two dataset, one with 60K records to train an ML module, other is 10K records to test the performance of the developed model. Actually it is a single 70K records dataset, split into two. Each record in this database consists of 28 by 28 pixel size digit image data and its corresponding label. 28 width * 28 height = 784 pixels represents the gray scale image capture of a single handwritten digit and an integer type label that represents one of the ten digits (from 0 to 9). So each row in the database consist of 785 columns where the first column is label, the rest 784 columns represent the pixel values of a handwritten digit

image. Each pixel value range is from 0 to 255 that corresponds to gray intensity (0 white, 255 black and all inbetween are gray levels). Below image shows first few rows and header of the database table.

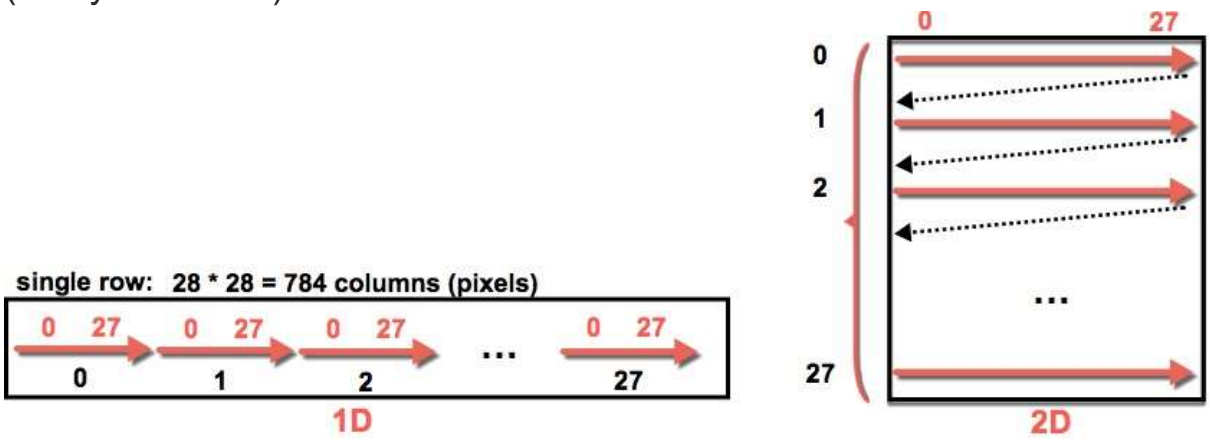


Label	f0	f1	f2	f3	f4	f5	f6	f7	f8	f9	f10	f11	f12	f13	...	f778	f779	f780	f781	f782	f783
7	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0

First row (excluding the header, column names) of the 10K dataset represents an image of handwritten digit 7. If we convert this row data into an black-white (all pixel values >0 is converted to 255) image representation, it will look like:



How we represent a one dimensional single row of data with 784 columns as 28 by 28 pixel size two dimensional image? Reshaping these 784 columns as 28 rows per line, we will get 28x28 pixel image (28 by 28 matrix).

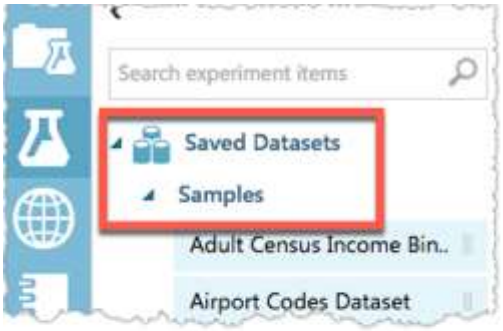


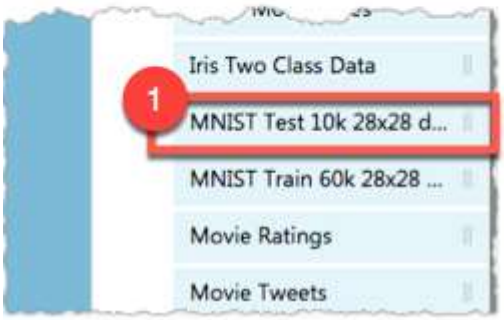
which is in numerical representation:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	
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	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
2	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	2
3	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	3
4	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	4
5	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	5
6	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	6
7	0	0	0	0	0	0	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	7
8	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	8
9	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	9
10	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	10
11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	11
12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	12
13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	13
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	14
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	15
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	16
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	17
18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	18
19	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	19
20	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	20
21	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	21
22	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	22
23	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	23
24	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	24
25	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	25
26	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	26
27	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	27
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	

Process MINST database in Azure ML with Python script

1. Drop "MNIST Test 10k 28x28 dense" module on the experiment canvas.





2. Drop an "Execute Python Script" module and set its "Python Script" property with following script. (fig. 3)

```
import pandas as pd
import numpy as np
from PIL import Image

def azureml_main(df1 = None, df2 = None):
    IMG_W = 28       # Image width in pixels
    IMG_H = 28       # Image height in pixels
    IMG_IDX_TOREAD = 0 # Read first image in the dataset

    # Dataframe to numpy array
    npa = df1.as_matrix()

    # Read raw digit data from column f0 to f800...
    dgtpx = npa[IMG_IDX_TOREAD, 1:]

    # convert img data to marix form
    dgtpx = np.reshape(dgtpx, (IMG_H, IMG_W)).astype('uint8')

    # We will make on/off pixel count
    # Convert to binary, 0 & 1 (just black & white)
    dgtpx[np.where(dgtpx != 0)] = 1

    # Image func. accepts data ranges between 0 - 255 (black & white)
    # Multply w 255 so we have just 0 & 255s (no gray scale)
    img = Image.fromarray(dgtpx * 255)
    img.save('digit.png')

    return pd.DataFrame(dgtpx),
```

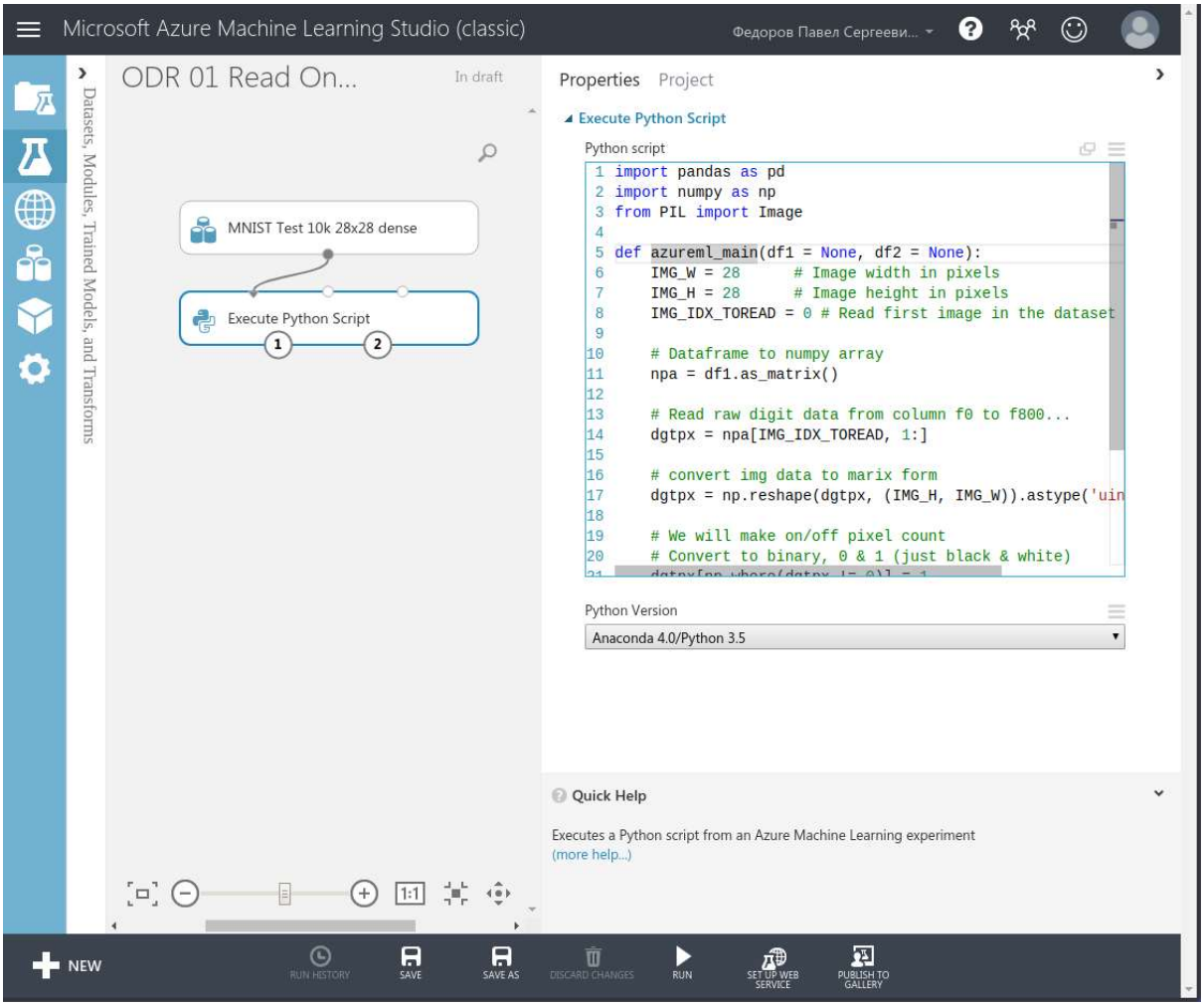


Fig. 3

3. Make connections and RUN the experiment. After execution finished, right click on the right output port (device output) of the "Execute Python Script" module and than click on "Visualize" menu item. (fig. 4)

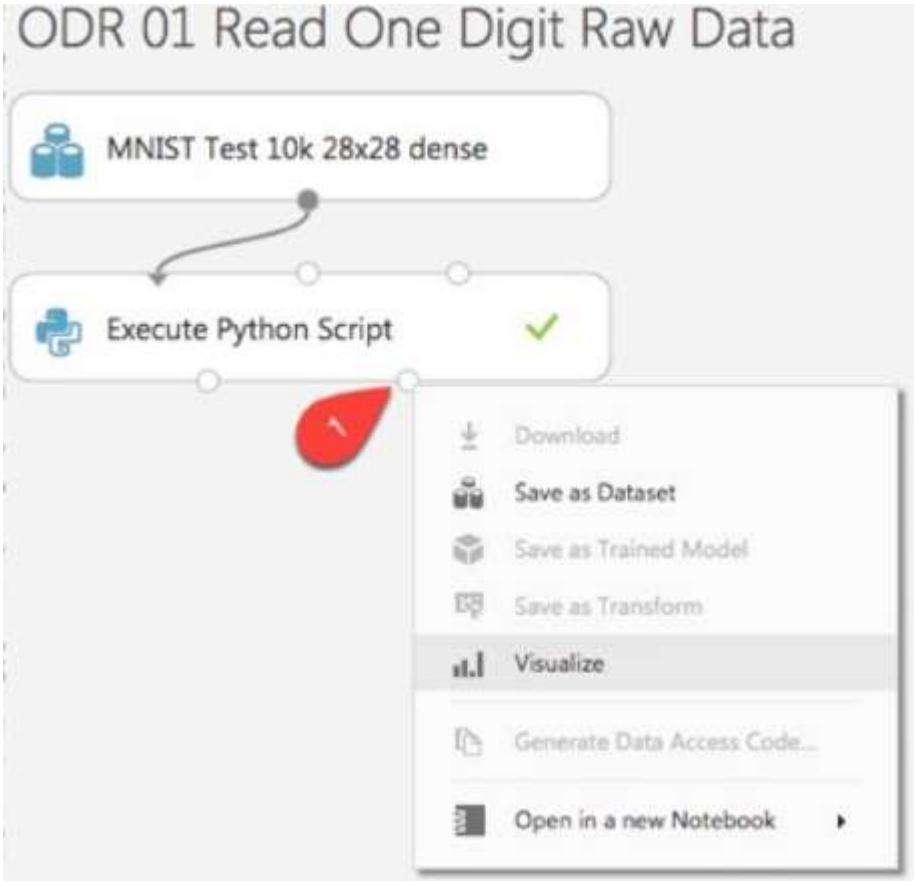


Fig. 4

4. On the output window, you will see the device output with saved image file. (fig. 5)

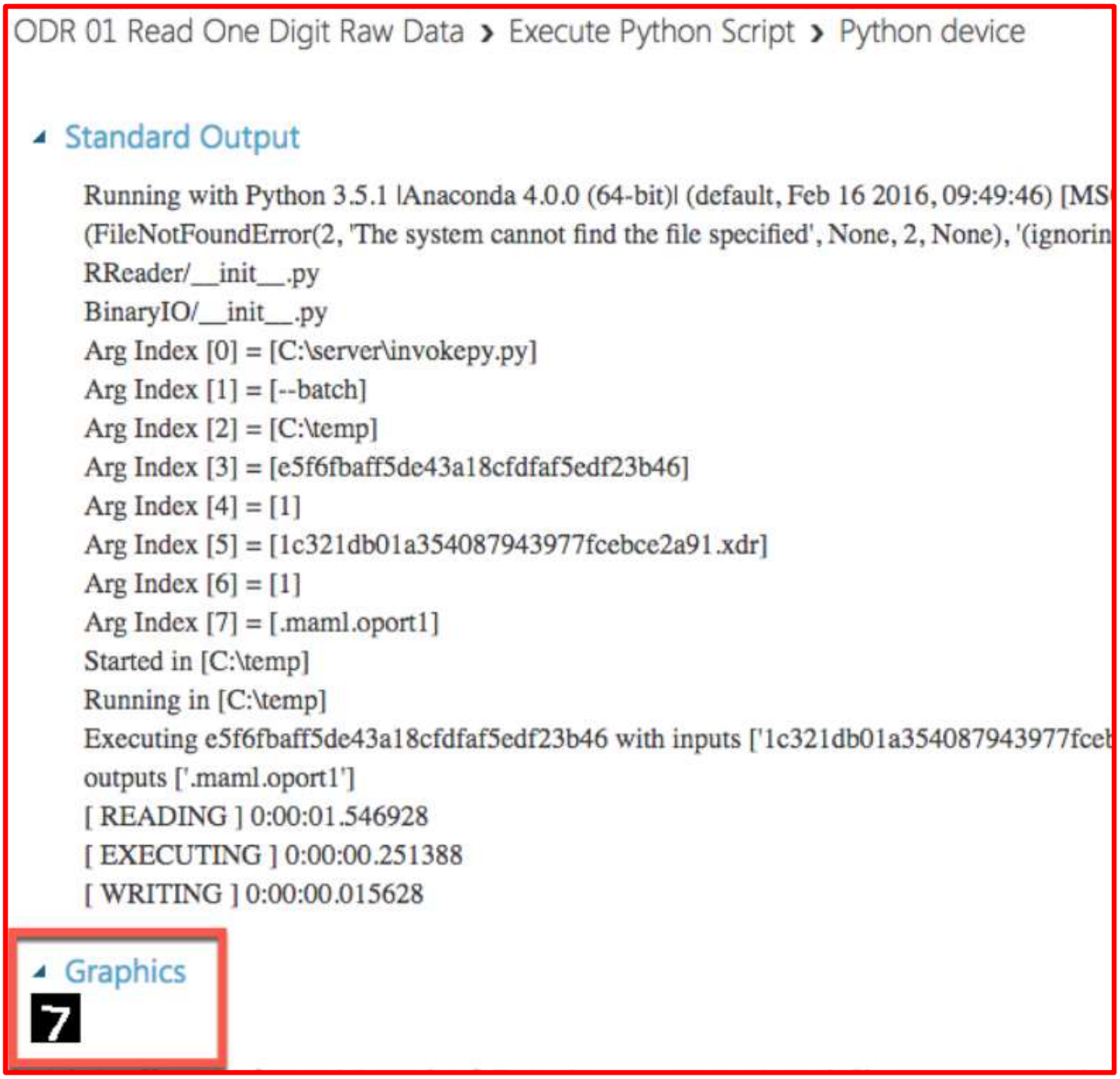


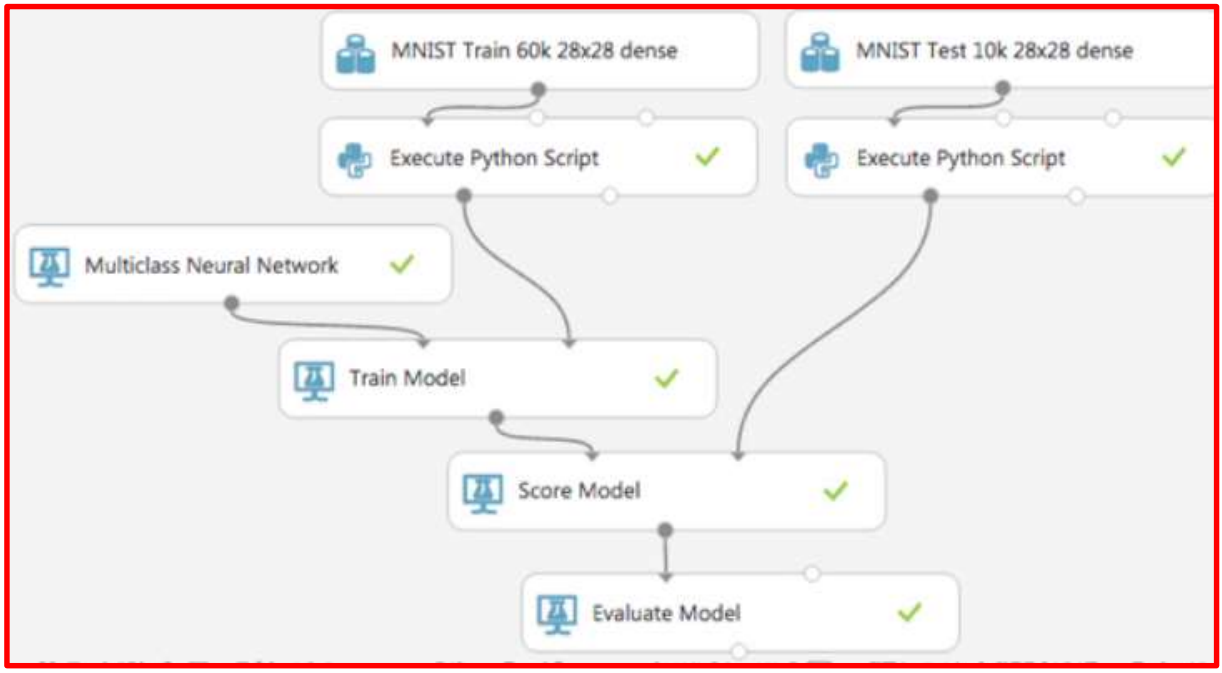
Fig. 5

Azure ML solution for OCR

Now we understand the contents of the database, how this handwritten digital images represented as pixels. In the following steps, we will develop an Azure ML solution to train a model to recognize any handwritten digital image.

Develop Azure ML experiment

1. Drop the following modules and make the appropriate connections (fig. 6)



(Fig. 6)

2. Set both "Execute Python Script" modules properties to the following script:

```
import pandas as pd
import numpy as np

def azureml_main(df1 = None, df2 = None):
    npa = df1.as_matrix()

    dgtpx = npa[:, 1:]

    dgtpx[np.where(dgtpx != 0)] = 1

    npa[:, 1:] = dgtpx

    result = pd.DataFrame(npa)

    result.columns = df1.columns.values

    return result,
```

This script will normalize the pixel values that range between 0 to 255 to 0 and 1. So values in 784 colums of the dataset will be either 0 or 1 (on or off pixel).

3. Set "Train Model" module's label property to "Label" with following steps.

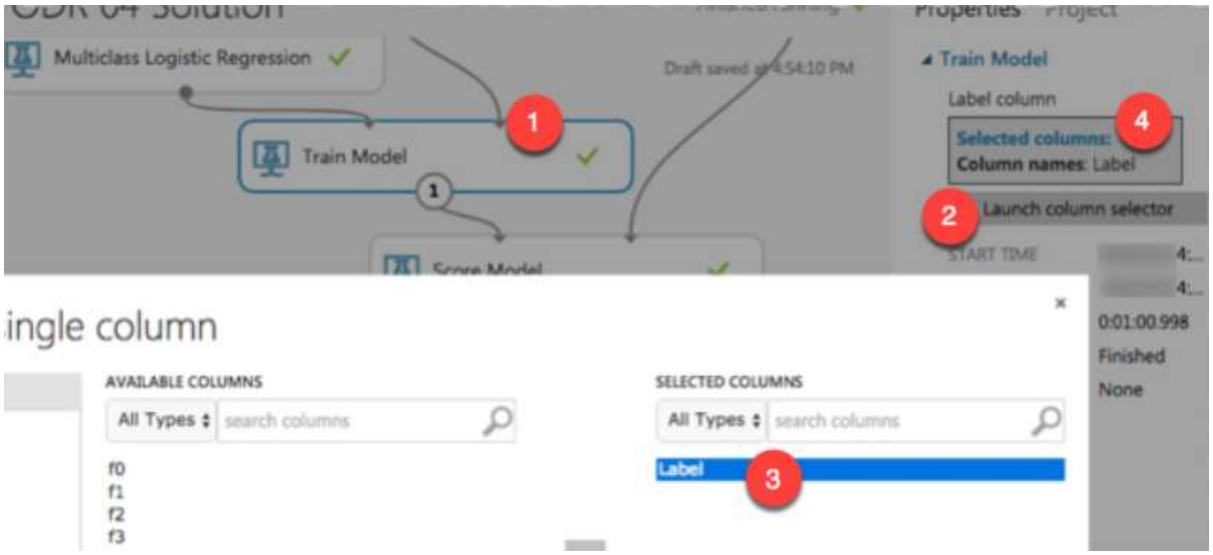


Fig. 7

4. In Multiclass Neural Network module set default params and random seed to 18
5. Run the experiment.
6. If you visualize the output port of the "Evaluate Model" module, you will see the precision of the model's prediction.

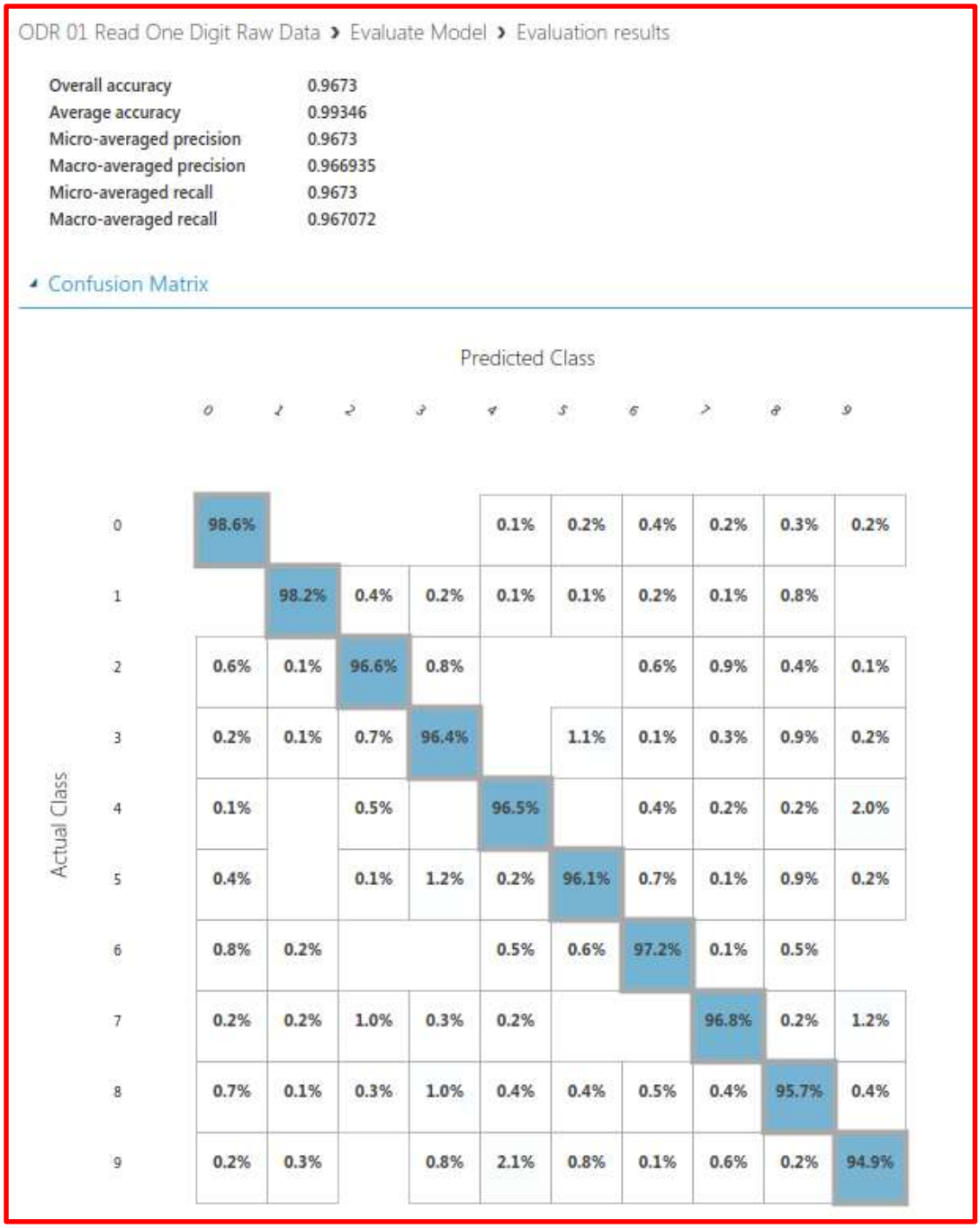


Fig. 8

as you can see from the matrix diagonal, model makes prediction almost with 90% precision, which is acceptable.

Deploy as web service

1. Click on "Set up Web Service" button on the bottom toolbar. (fig. 9)

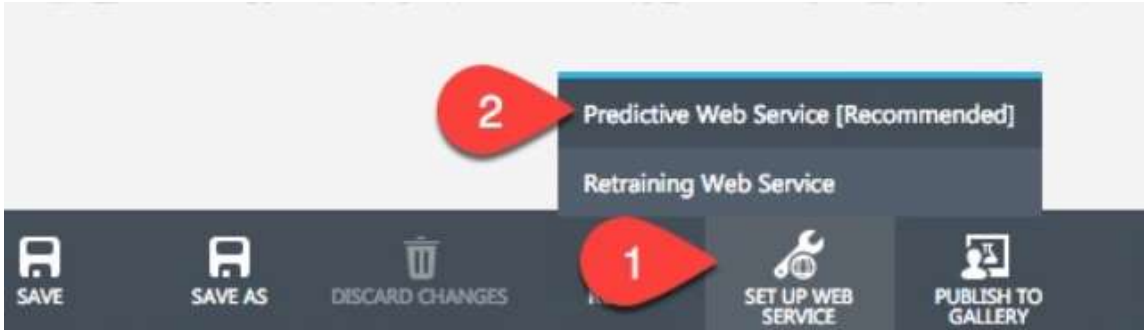


Fig. 9

2. Click on "Predictive Web Service [Recommended]" item on the popup menu.
3. On the "Predictive experiment" canvas, drag&drop two "Select Columns in Dataset" modules on the canvas. (fig. 10)

Fig. 10

4. Connect the I/O ports of the "Select Columns in Dataset" modules as shown on the above image.

5. For the first "Select Columns in Dataset" module, click on the "Launch Column Selector" button on the properties window. Select 784 of the 785 column names (except the column named "Label"). And add them to the right bucket. Click "OK" button on the left bottom corner.

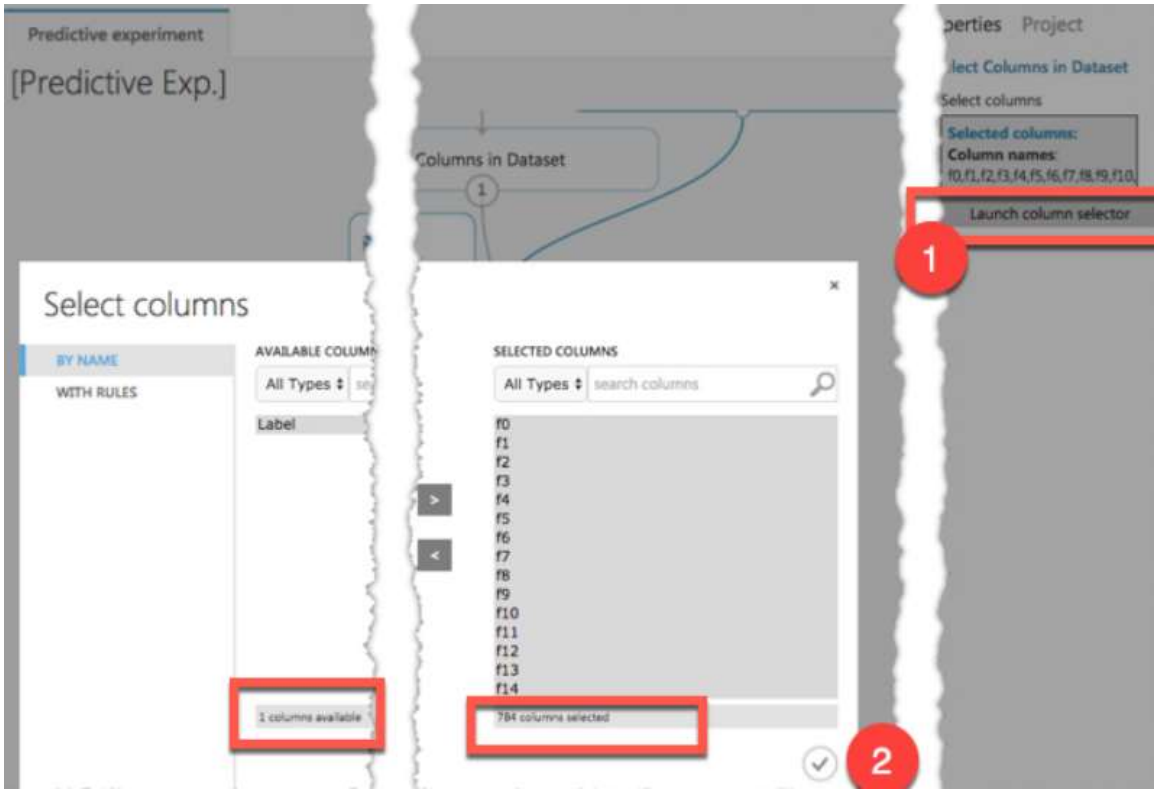


Fig. 11

6. For the second "Select Columns in Dataset" module, click on the "Launch Column Selector" button on the properties window. Select just the column named "Scored Labels". And add it to the right bucket. Click "OK" button on the left bottom corner.
7. Save and then Run the experiment. (fig. 12)

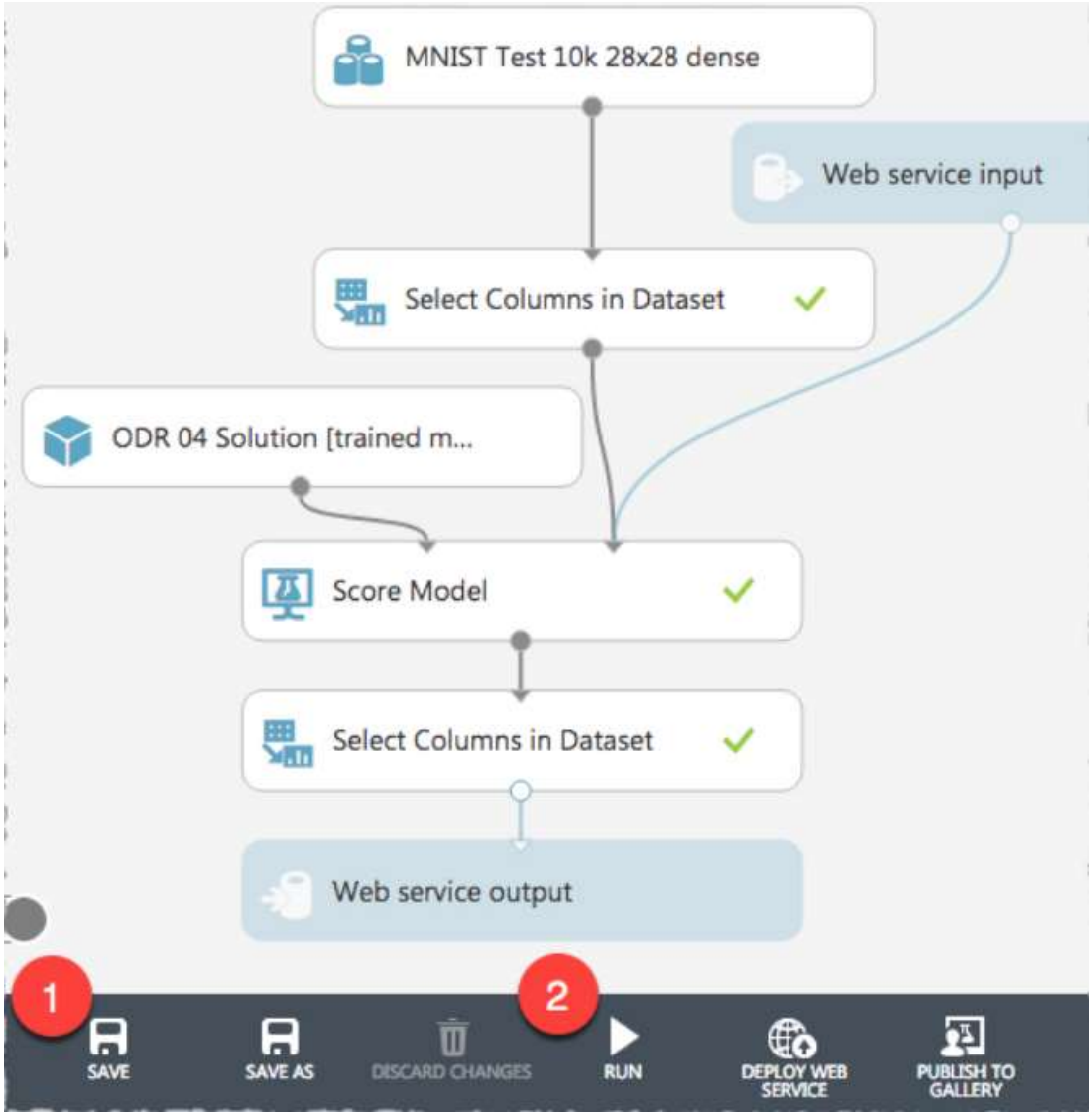


Fig. 12

1. Now click on "Deploy Web Service [Classic]" menu item under "Deploy Web Service". (just only deploy web service, without choice of new or classic) (fig. 13)

Fig. 13

Parameters needed to publish with management API

Take a note about the below three parameters of the published web service in the previous step. We will use them in the next section

1. Note the "API Key" (1st parameter to note) on the WebService dashboard page. (fig. 14)

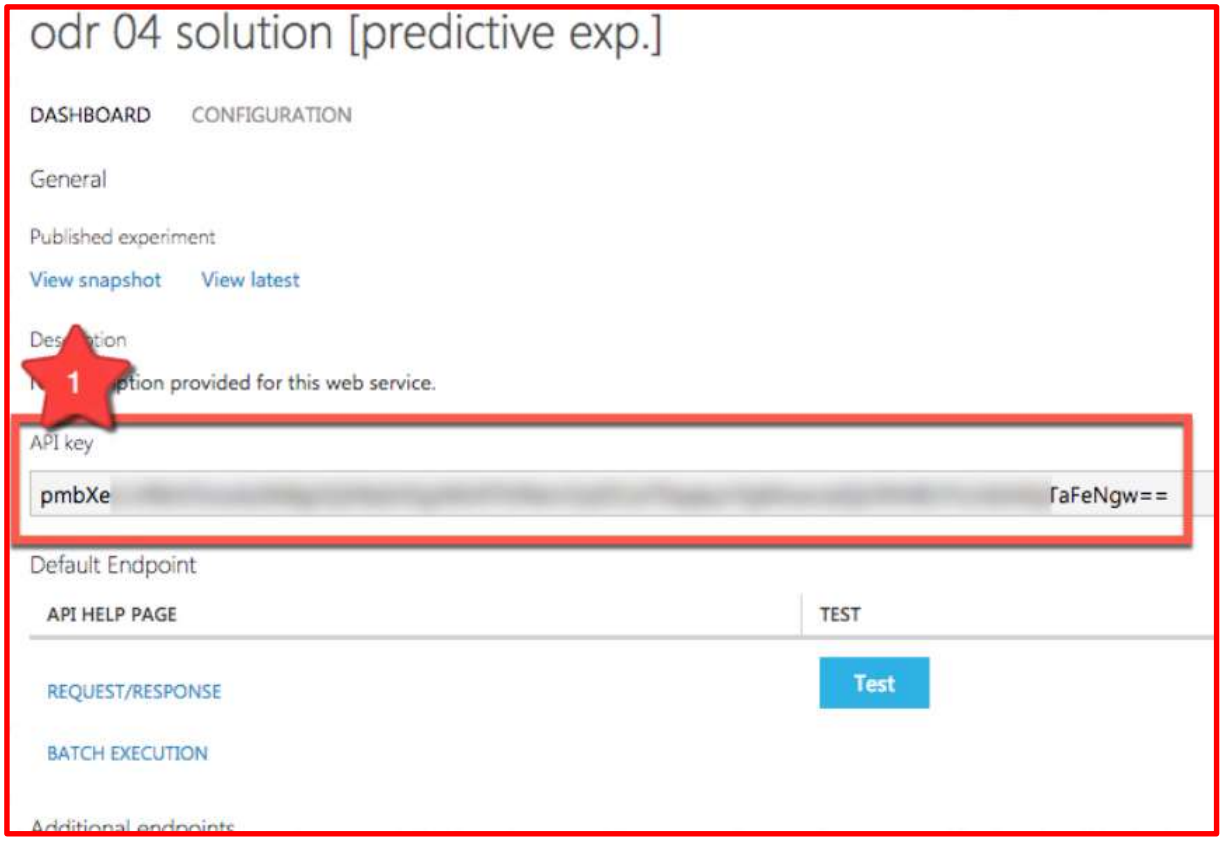


Fig. 14

2. Click on the "Request/Response" link on the same page (page in the prev. step).

3. Note the "Request URI" (2nd parameter to note) on the "Request/Response" dashboard page. (fig. 15)

Fig. 15

On the same page, scroll down and note the Sample Request (3rd parameter to note) code. (in this case, very long lines of code with 785 parameters.) (fig. 16)

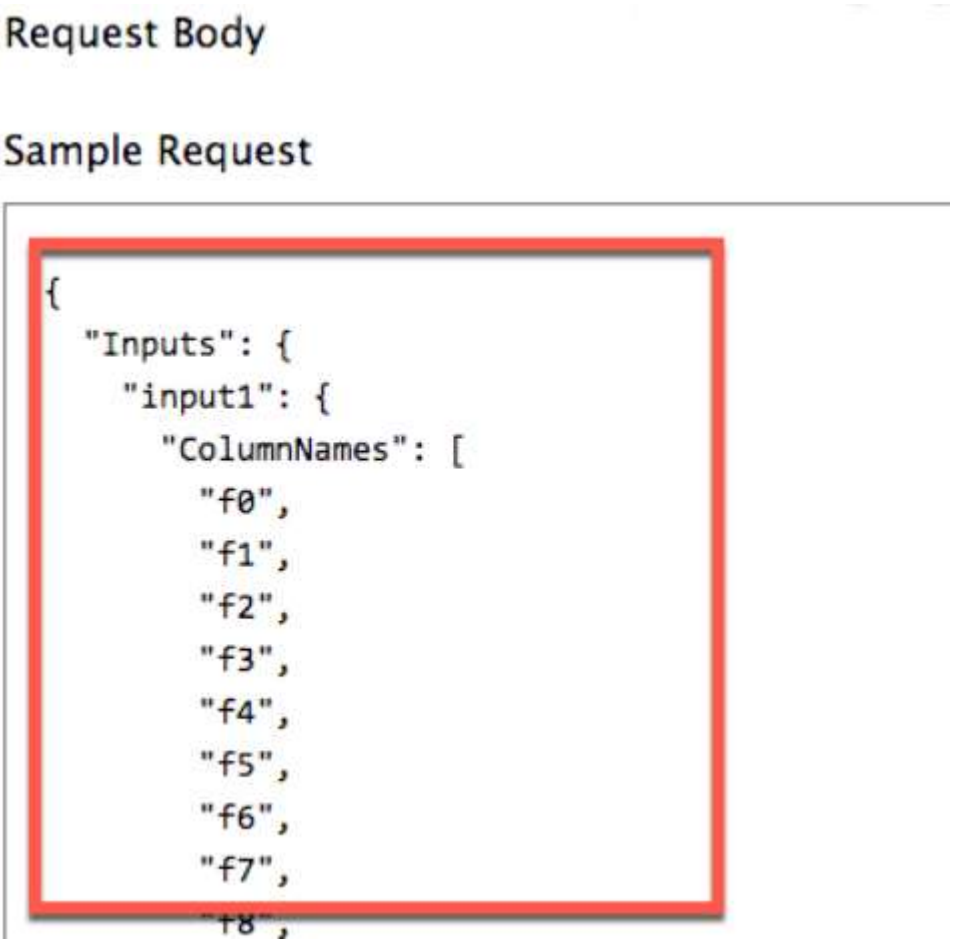


Fig. 16

Consuming the ML solution

Now, using the management API service (https://portal.azure.com/#home), we will create a public, access controlled, CORS enabled endpoint for our ML Web service that we created in the prev. stage, but first of all you should activate students subscription(https://azure.microsoft.com/ru-ru/free/students/?cdn=disable). After making it, you should find API management service in the search window.

1. Create a new API Management service under the Azure management portal (browse api management services). (fig. 17)

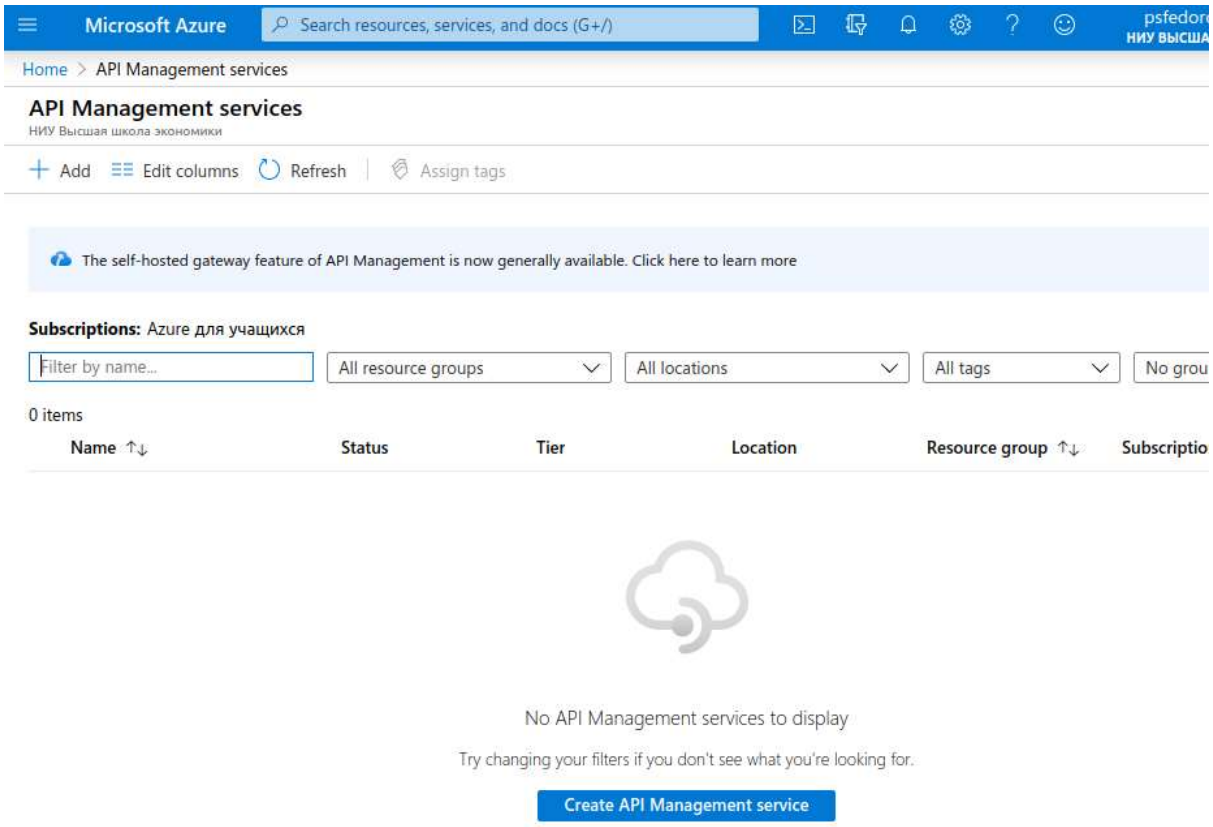


Fig. 17

2. Give a unique URL name and select a service region then click next (right arrow) and enter any organization name and your contact email address and click OK. (fig. 19)

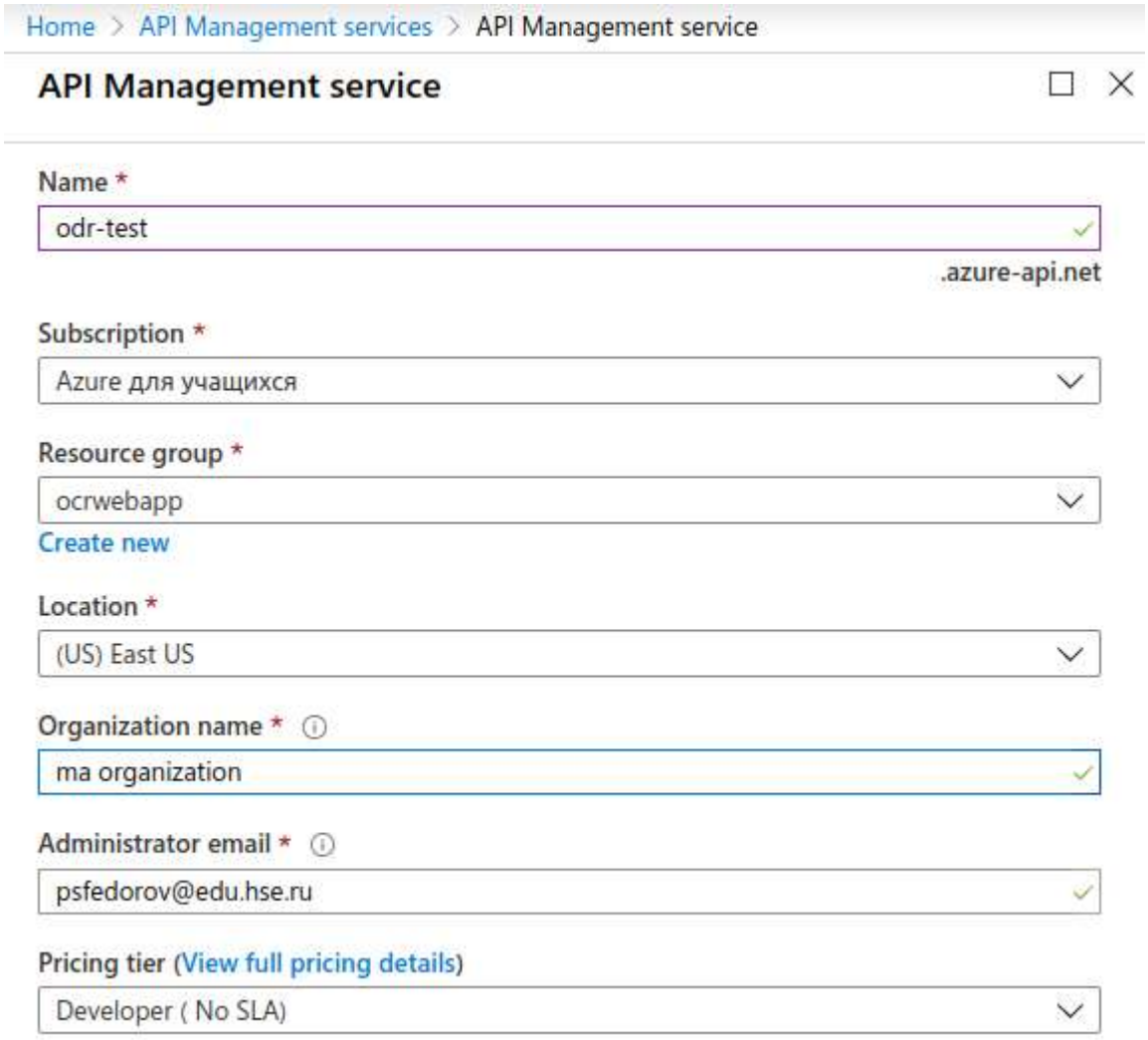


Fig. 19

3. Azure will need some time to create it, app. 25-30 minutes. Once the service created and the status is ready, click on the ‘your service name’ butto. (fig. 20)

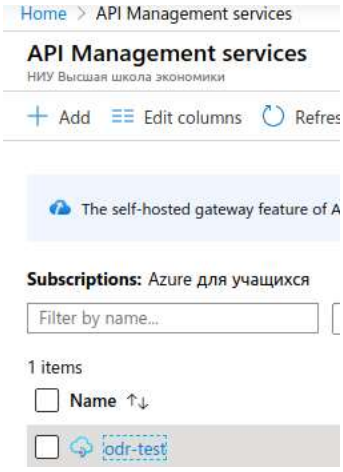


Fig. 20

4. On the management page, switch to API tab and click "Add API" (fig. 21)

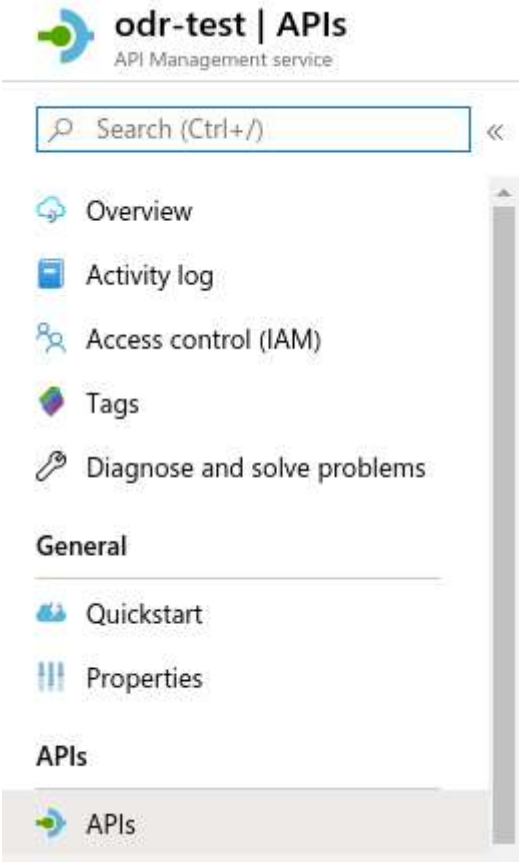


Fig. 21

5. Fill in the create API form. You can write any descriptive name to your API in the Web API name field. In this API namespace (https://odr.azure-api.net/) there may be more than one service. To identify this specific service give any name as a suffix in the Web API URL suffix field. In the Web service URL field copy and paste the Request URI value that you noted earlier. But do not paste the suffix part of the Request URI. i.e. if the URI is:
https://ussouthcentral.services.azureml.net/workspaces/4714.....bcc7fa7d7/execute?api-version=2.0&details=true
then just copy the following part:
https://ussouthcentral.services.azureml.net/workspaces/4714.....bcc7fa7d7
we will use the below suffix part in the following stages.
/execute?api-version=2.0&details=true
after all, you can press the save button to create the API and goto the operations page of the API. (fig. 22)

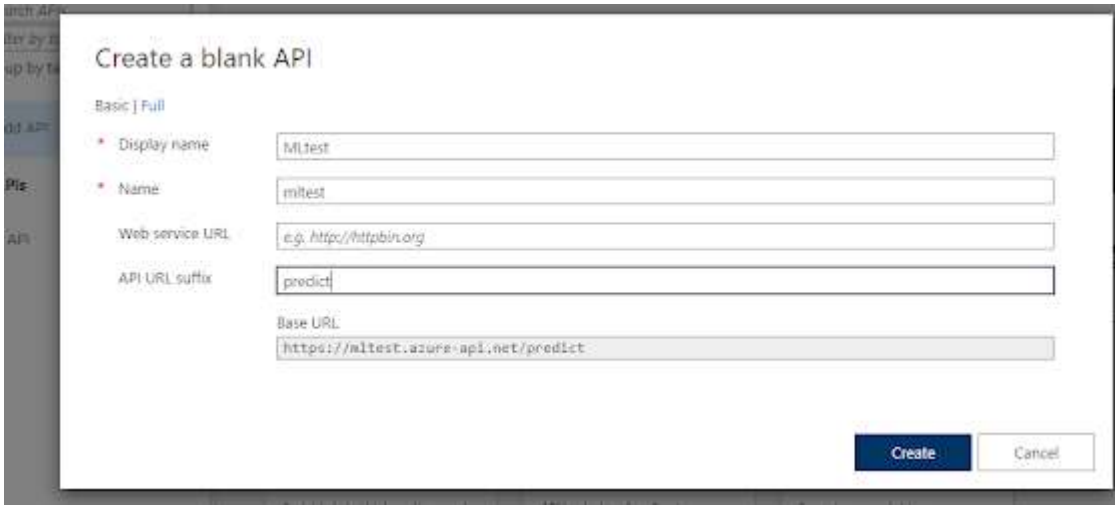


Fig. 22

6. On the Operations page, click on the Add Operation link. (fig. 23)

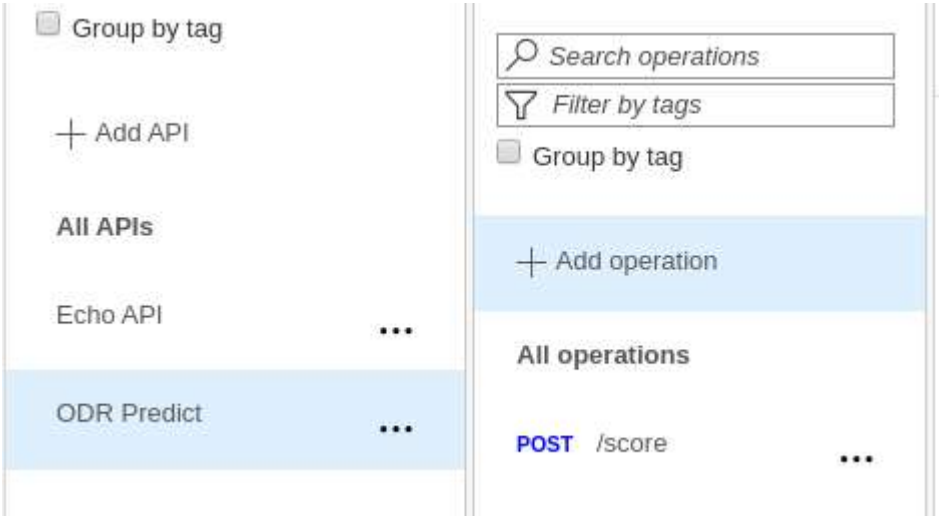


Fig. 23

7. On the Signature tab of the operations page select POST method in the HTTP web field. Enter any URL template (in this sample we will use /score) that will be replaced with the suffix */execute?api-version=2.0&details=true* we noted in the previous step. (fig. 24)

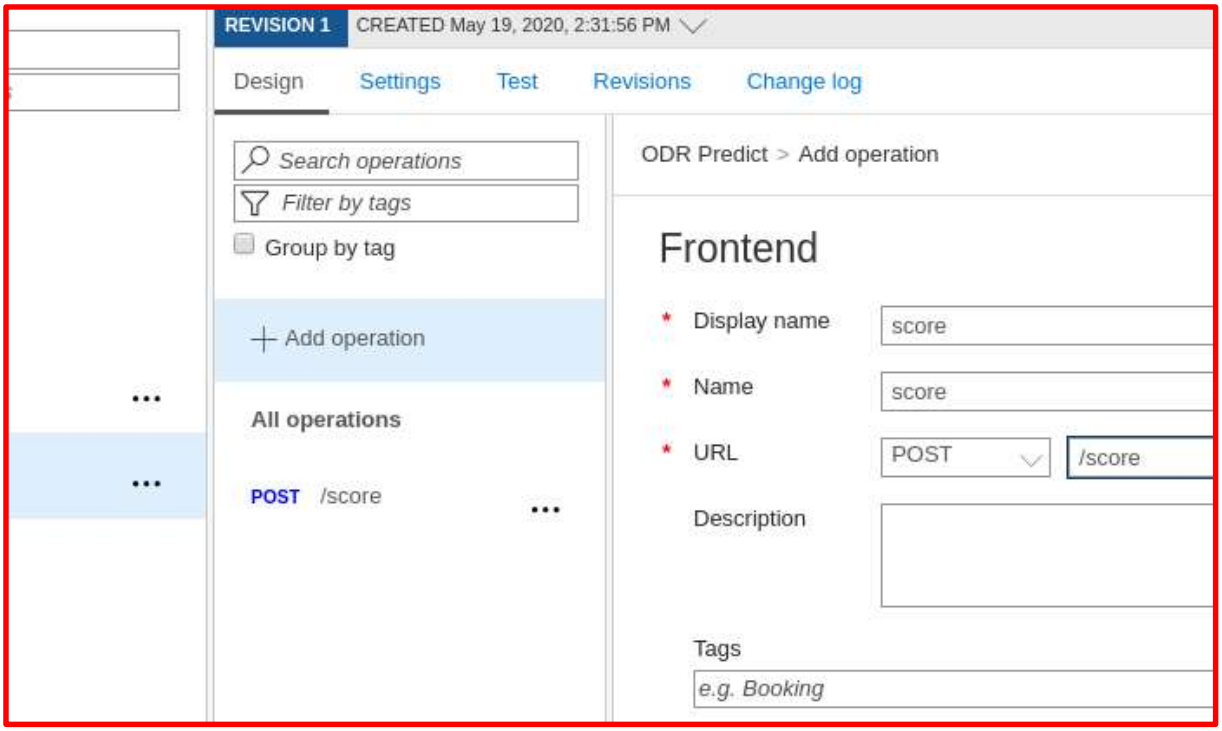


Fig. 24

8. In the *Body* page of the *Operations* tab of the API, click on the Add Representation link. (fig. 25)

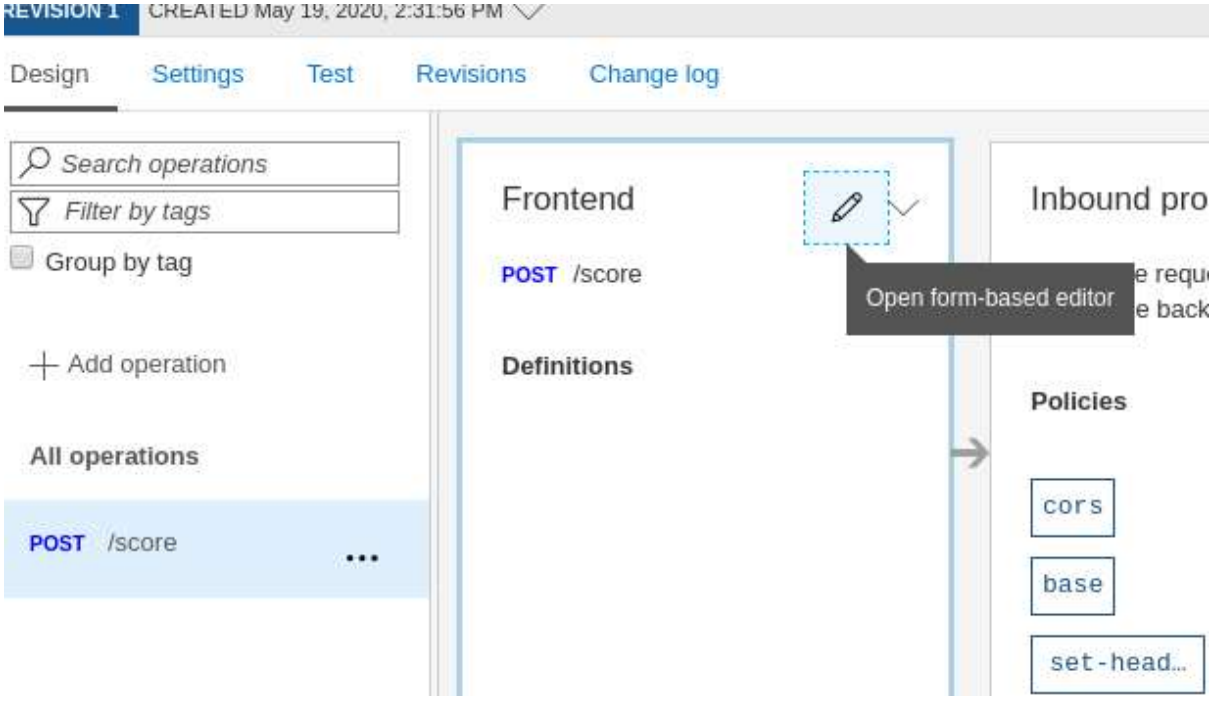


Fig. 25

9. Here type application/json and press *Enter* (fig. 26)

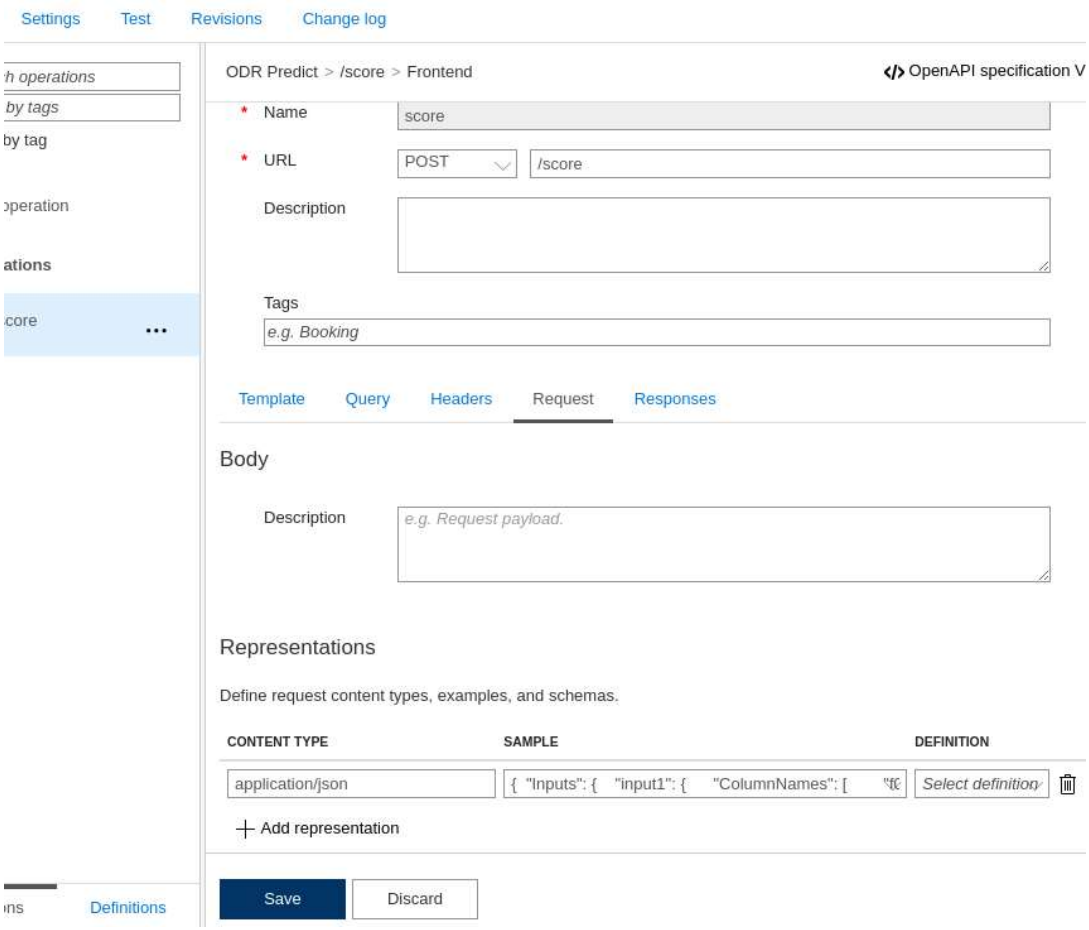


Fig. 26

10. In the Representation example field paste the Sample Request value that we noted in the above section (fig. 27)

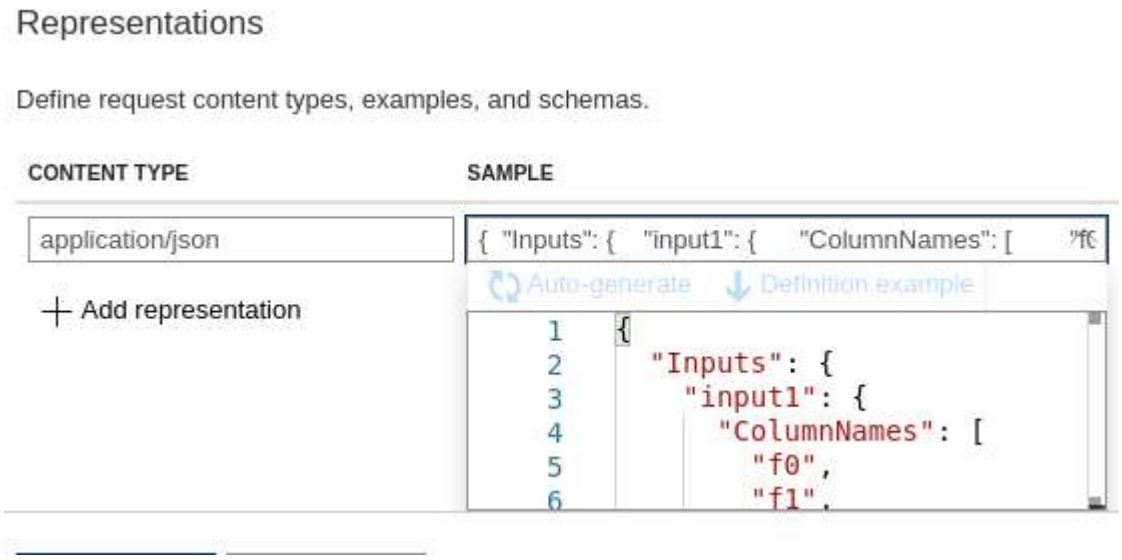


Fig. 27

11. Now switch to the Policies tab in the *API Management* page. Select Predict API and the /score operation in the combo boxes.

Finally click on the Configure Policy link (code editor in backend Policies). On this page you can do lots of fun staff to customize your API. (fig. 28)

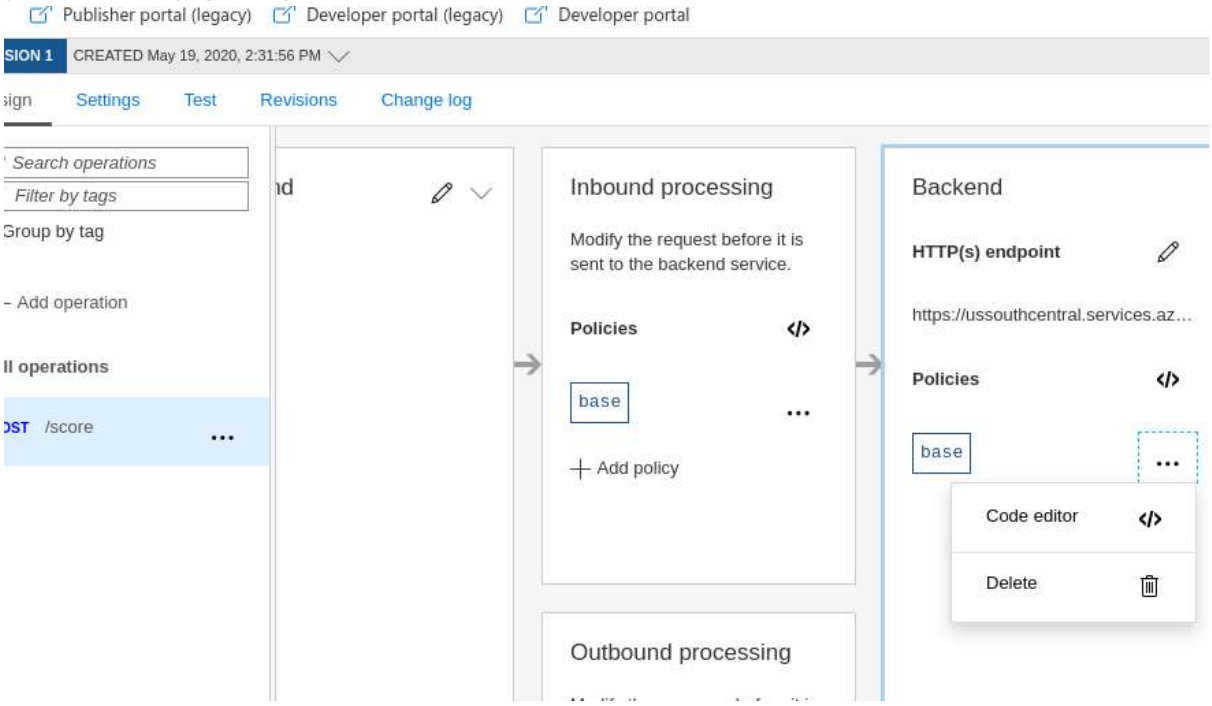


Fig. 28

12. After you click on the Configure Policy link, the *Policy definition* field will become editable.
Press *Enter* key to create a blank line under <inbound><base/> tags as shown in the below screenshot.

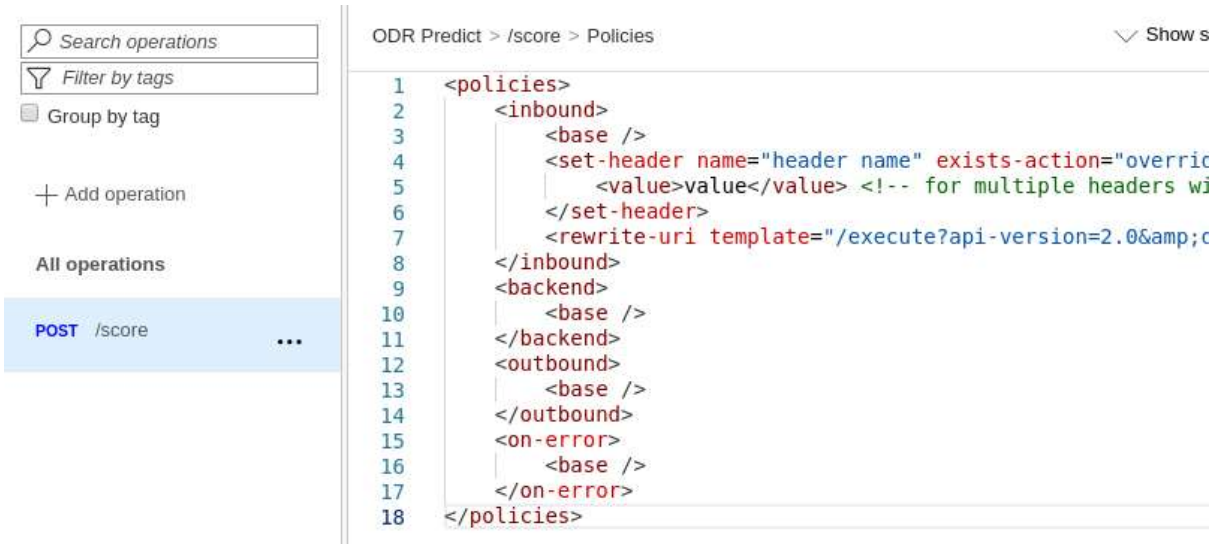


Fig. 29

13. After adding the template, it will look like:

```
<policies>
    <inbound>
        <base />
        <set-header name="header name" exists-action="override | skip |
append | delete">
            <value>value</value> <!-- for multiple headers with the same
name add additional value elements -->
        </set-header>
        <rewrite-uri template="/execute?api-version=2.0&amp;details=true"
/>
    </inbound>
    <backend>
        <base />
    </backend>
    <outbound>
        <base />
    </outbound>
    <on-error>
        <base />
    </on-error>
</policies>
```

14. Update the above policy template with the API Key that you noted. Your API Key should be typed immediately after the Bearer keyword with a preciding space character as shown below. so it will look like as (Below sample uses random API Key):

```
<policies>
    <inbound>
        <base />
        <set-header name="Authorization" exists-action="override">
            <value>Bearer wJJx5JxpO6C......Tey2Zu/tzJpo+p5DWRg==</value>
<!-- for multiple headers with the same name add additional value elements
-->
        </set-header>
        <rewrite-uri template="/execute?api-version=2.0&amp;details=true"
/>
    </inbound>
    <backend>
        <base />
    </backend>
    <outbound>
        <base />
    </outbound>
    <on-error>
        <base />
    </on-error>
</policies>
```

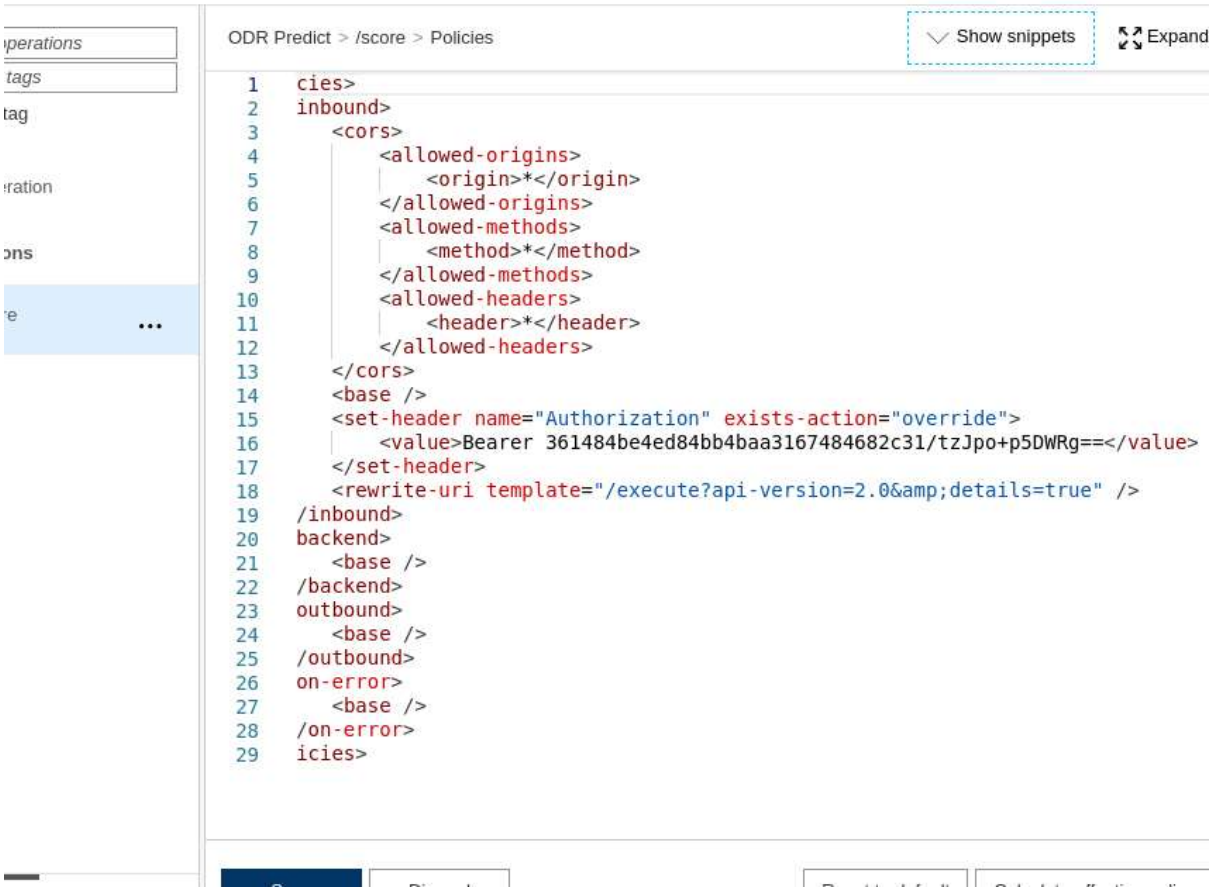


Fig. 30

15. Make the last update by adding the following lines under <inbound><base/> tag. You can either enter manually or use the *CORS* item under policy statements to insert it. (fig. 31)

```
<cors>
    <allowed-origins>
        <origin>*</origin>
    </allowed-origins>
    <allowed-methods>
        <method>*</method>
    </allowed-methods>
    <allowed-headers>
        <header>*</header>
    </allowed-headers>
</cors>
```

This will allow any method, any IP address to access from cross origin. Above settings doesn't have any restriction and generally you have to make modification on it to have secure web apps. Refer to the following address to read more details on policy settings. And don't forget to add service URL at the HTTPS endpoint section Backend's, that we noted when we've deployed our ML solution

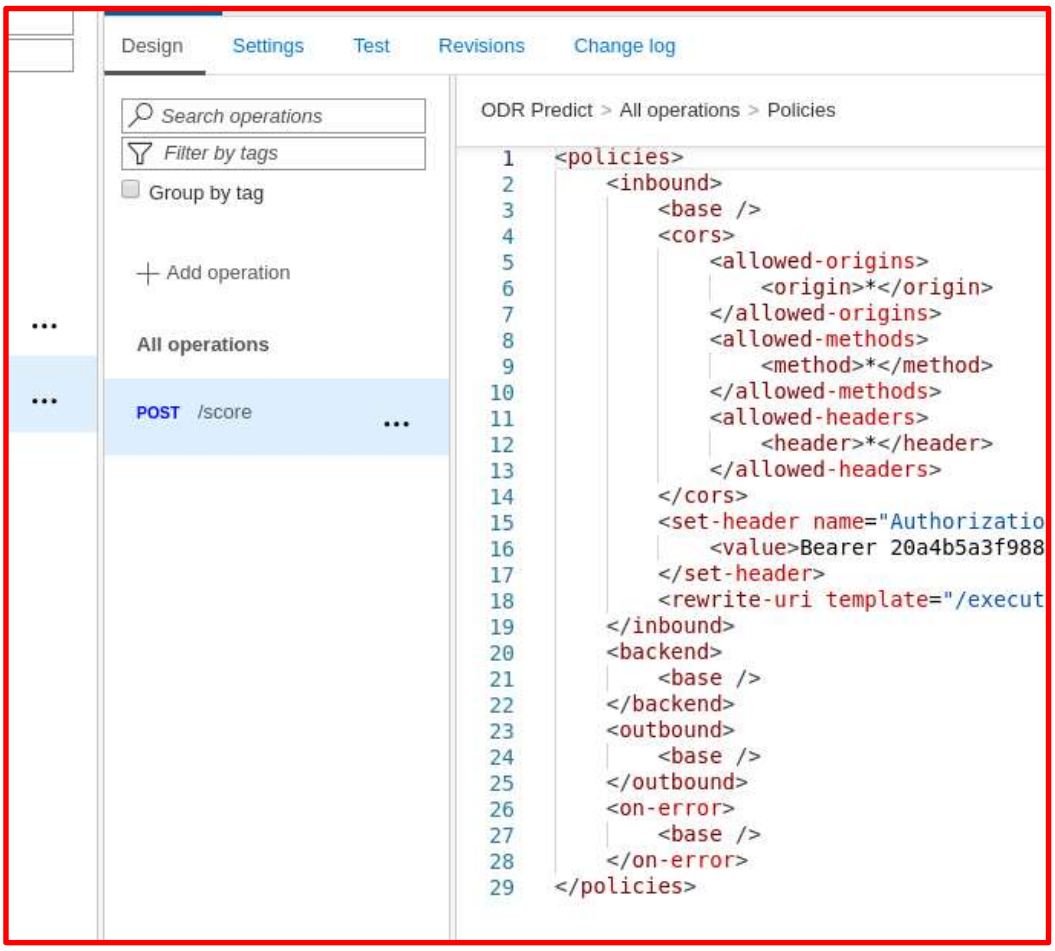
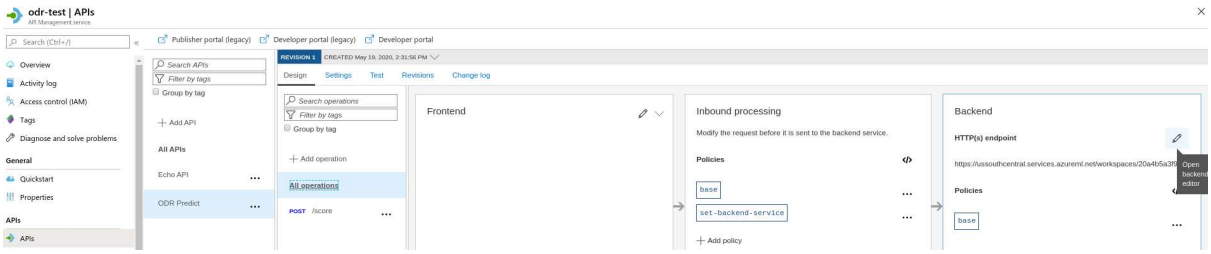
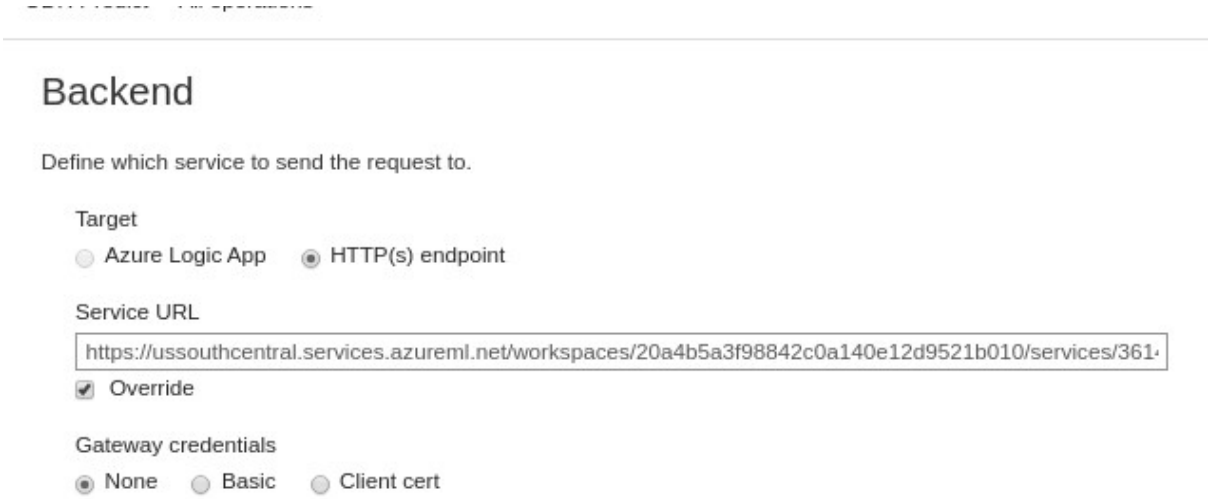


Fig. 31

16. Finally click the Save button to save all these changes and finalize our API settings.
17. And you must click override your service url, like on picture below.





Security of the API

For simplicity, we will let anyone to access our service. If you want to set authentication or call limit etc. to this web service, you can add as much user as you want through the users/security tab and use the user specific keys (passwords) to access the web service.

1. Open the security tab. Set "Proxy authentication" and "User authorization" properties to "None" and save the state. (fig. 32)

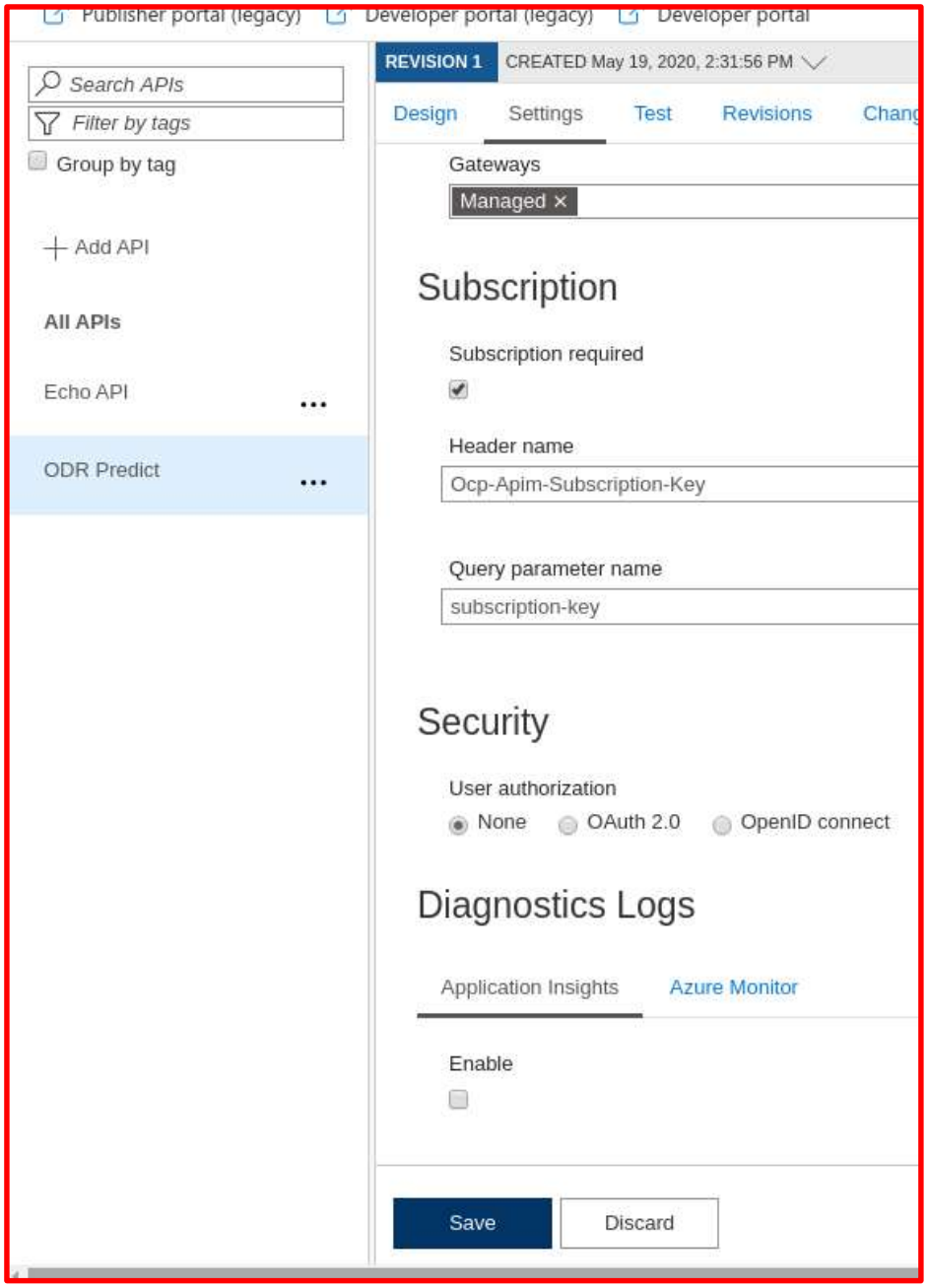


Fig. 32

2. Now, anyone, without authentication (highly recommended to set user authentication, see) can access our webservice through the url address "https://odr.azure-api.net/predict/score" that was shown at earlier.

Develop web application

In this step we will develop a web application (static webpage with javascript) that will use the web service created and published through previous steps.

1. Write the following html code inside the "index.html" file and save it.

```
<html>
<head>
    <title>Optical Character Recognition with Microsoft Azure Machine Learning</title>
    <style>
        #imgView {
            border: 1px solid #FFFFFF;
        }

        #btnClear {
            background: gray;
            width: 280px;
            color: white;
            font-size: 2em;
        }

        #btnSend {
            background: gray;
            width: 280px;
            color: white;
            font-size: 2em;
        }
    </style>
</head>

<body>
    <canvas id="imgView" width="280" height="280">
        Unfortunately, your browser does not supported.
    </canvas>

    <p><button id="btnClear">Clear</button></p>
    <p><button id="btnSend">Send</button></p>

    <script src="https://ajax.googleapis.com/ajax/libs/jquery/1.9.1/jquery.min.js"></script>
    <script src="./odr.js"></script>
</body>
</html>
```

Now create another file (our javascript file that will do all the work) named "odr.js" and save it in the same folder "ocrweb". odr.js name is statically embedded into html file so any name change should be reflected there too. (In line 143 paste your API's url)

2. Set the content of the odr.js file as:

```
window.addEventListener('load', function () {
    var canvas, context;
    var iw = 28; // image width
    var ih = 28; // image height
    var s = 10;  // scale (in HTML the canvas size is 280px, mapping canvas size to MINST sample size 28x28)
    var dimg;    // digit pixels in 2D array

    function init() {
        canvas = document.getElementById('imgView');
        context = canvas.getContext('2d');
        drawtool = new canvasPencil();

        canvas.addEventListener('mousedown', drawtool.mousedown, false);
        canvas.addEventListener('mousemove', drawtool.mousemove, false);
        canvas.addEventListener('mouseup', drawtool.mouseup, false);
        canvas.addEventListener('mouseleave', drawtool.mouseup, false);

        var btnClear = document.getElementById("btnClear");
        btnClear.addEventListener("click", clearCanvas, false);

        var btnSend = document.getElementById("btnSend");
        btnSend.addEventListener("click", callWebService, false);

        drawCanvasGrid();

        dimg = new Array(ih);
        for (var i = 0; i < ih; i++) {
            dimg[i] = new Array(iw);
        }

        clearCanvas();
    }

    function canvasPencil() {
        var isMouseDown = false;
```

```
36.         var mouseX = 0;
37.         var mouseY = 0;
38.
39.         this.mousedown = function (evt) {
40.             isMouseDown = true;
41.
42.             mouseX = evt.offsetX;
43.             mouseY = evt.offsetY;
44.
45.             context.beginPath();
46.             context.moveTo(mouseX, mouseY);
47.         };
48.
49.         this.mousemove = function (evt) {
50.             if (isMouseDown) {
51.                 mouseX = evt.offsetX;
52.                 mouseY = evt.offsetY;
53.
54.                 dx = Math.floor(mouseX / 10);
55.                 dy = Math.floor(mouseY / 10);
56.
57.                 drawCanvasCell(dx, dy); // draw pixel on the canvas
58.
59.                 dimg[dx][dy] = 1;  // set the same pixel on 2D array
60.
61.
62.                 // not a thin line, but a bold line, like the original digit
    drawings...
63.                 if (dx < 27 && dy < 27 && dx > 0 && dy > 0){
64.                     drawCanvasCell(dx, dy + 1);
65.                     drawCanvasCell(dx, dy - 1);
66.                     drawCanvasCell(dx + 1, dy + 1);
67.                     drawCanvasCell(dx + 1, dy);
68.                     drawCanvasCell(dx + 1, dy - 1);
69.                     drawCanvasCell(dx - 1, dy + 1);
70.                     drawCanvasCell(dx - 1, dy);
71.                     drawCanvasCell(dx - 1, dy - 1);
72.
73.                     // not a thin line, but a bold line, like the original
    digit drawings...
74.                     dimg[dx][dy + 1]         = 1;
75.                     dimg[dx][dy - 1]         = 1;
76.                     dimg[dx + 1][dy + 1]     = 1;
77.                     dimg[dx + 1][dy]         = 1;
78.                     dimg[dx + 1][dy - 1]     = 1;
```

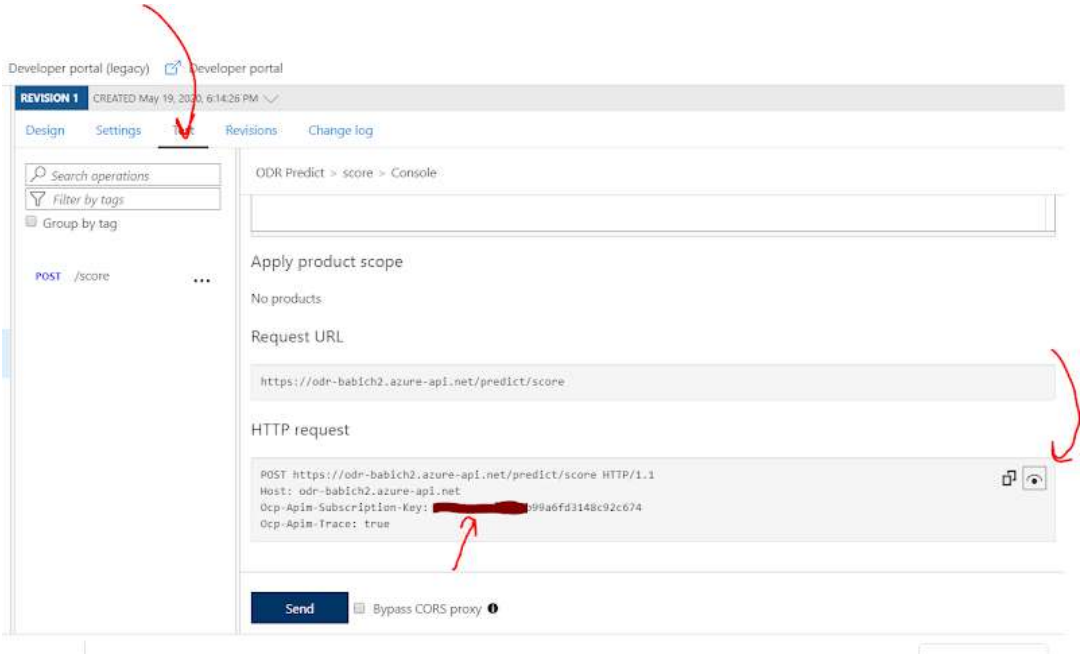
```
                    dimg[dx - 1][dy + 1]    = 1;
                    dimg[dx - 1][dy]        = 1;
                    dimg[dx - 1][dy - 1]    = 1;
                }
            }
        };

        this.mouseup = function (evt) {
            isMouseDown = false;
        };
    }

    function clearCanvas() {
        canvas.width = canvas.width;
        drawCanvasGrid();

        for (var i = 0; i < ih; i++) {
            for (var j = 0; j < iw; j++) {
                dimg[i][j] = 0;
            }
        }
    }

    function drawCanvasCell(x, y) {
        context.fillRect(x * s, y * s, s, s);
    }

    function drawCanvasGrid() {
        for (var x = 0; x <= iw; x += 1) {
            context.moveTo(x * s, 0);
            context.lineTo(x * s, ih * s);
        }

        for (var y = 0; y <= ih; y += 1) {
            context.moveTo(0, y * s);
            context.lineTo(iw * s, y * s);
        }

        context.stroke();
    }

    function callWebService() {
        var dimgarray = ""
        for (var i = 0; i < ih; i++) {
                for (var j = 0; j < iw; j++) {
```

```
                    dimgarray += dimg[j][i] + ",";
                }
            }
        dimgarray = dimgarray.slice(0, dimgarray.length - 1);

        var arg = {
                    "Inputs": {
                        "input1": {
                        "ColumnNames": ["f0", "f1", "f2", "f3", "f4", "f5",
"f6", "f7", "f8", "f9", "f10", "f11", "f12", "f13", "f14", "f15", "f16",
"f17", "f18", "f19", "f20", "f21", "f22", "f23", "f24", "f25", "f26", "f27",
"f28", "f29", "f30", "f31", "f32", "f33", "f34", "f35", "f36", "f37", "f38",
"f39", "f40", "f41", "f42", "f43", "f44", "f45", "f46", "f47", "f48", "f49",
"f50", "f51", "f52", "f53", "f54", "f55", "f56", "f57", "f58", "f59", "f60",
"f61", "f62", "f63", "f64", "f65", "f66", "f67", "f68", "f69", "f70", "f71",
"f72", "f73", "f74", "f75", "f76", "f77", "f78", "f79", "f80", "f81", "f82",
"f83", "f84", "f85", "f86", "f87", "f88", "f89", "f90", "f91", "f92", "f93",
"f94", "f95", "f96", "f97", "f98", "f99", "f100", "f101", "f102", "f103",
"f104", "f105", "f106", "f107", "f108", "f109", "f110", "f111", "f112",
"f113", "f114", "f115", "f116", "f117", "f118", "f119", "f120", "f121",
"f122", "f123", "f124", "f125", "f126", "f127", "f128", "f129", "f130",
"f131", "f132", "f133", "f134", "f135", "f136", "f137", "f138", "f139",
"f140", "f141", "f142", "f143", "f144", "f145", "f146", "f147", "f148",
"f149", "f150", "f151", "f152", "f153", "f154", "f155", "f156", "f157",
"f158", "f159", "f160", "f161", "f162", "f163", "f164", "f165", "f166",
"f167", "f168", "f169", "f170", "f171", "f172", "f173", "f174", "f175",
"f176", "f177", "f178", "f179", "f180", "f181", "f182", "f183", "f184",
"f185", "f186", "f187", "f188", "f189", "f190", "f191", "f192", "f193",
"f194", "f195", "f196", "f197", "f198", "f199", "f200", "f201", "f202",
"f203", "f204", "f205", "f206", "f207", "f208", "f209", "f210", "f211",
"f212", "f213", "f214", "f215", "f216", "f217", "f218", "f219", "f220",
"f221", "f222", "f223", "f224", "f225", "f226", "f227", "f228", "f229",
"f230", "f231", "f232", "f233", "f234", "f235", "f236", "f237", "f238",
"f239", "f240", "f241", "f242", "f243", "f244", "f245", "f246", "f247",
"f248", "f249", "f250", "f251", "f252", "f253", "f254", "f255", "f256",
"f257", "f258", "f259", "f260", "f261", "f262", "f263", "f264", "f265",
"f266", "f267", "f268", "f269", "f270", "f271", "f272", "f273", "f274",
"f275", "f276", "f277", "f278", "f279", "f280", "f281", "f282", "f283",
"f284", "f285", "f286", "f287", "f288", "f289", "f290", "f291", "f292",
"f293", "f294", "f295", "f296", "f297", "f298", "f299", "f300", "f301",
"f302", "f303", "f304", "f305", "f306", "f307", "f308", "f309", "f310",
"f311", "f312", "f313", "f314", "f315", "f316", "f317", "f318", "f319",
"f320", "f321", "f322", "f323", "f324", "f325", "f326", "f327", "f328",
"f329", "f330", "f331", "f332", "f333", "f334", "f335", "f336", "f337",
"f338", "f339", "f340", "f341", "f342", "f343", "f344", "f345", "f346",
"f347", "f348", "f349", "f350", "f351", "f352", "f353", "f354", "f355",
"f356", "f357", "f358", "f359", "f360", "f361", "f362", "f363", "f364",
```

"f365", "f366", "f367", "f368", "f369", "f370", "f371", "f372", "f373",
"f374", "f375", "f376", "f377", "f378", "f379", "f380", "f381", "f382",
"f383", "f384", "f385", "f386", "f387", "f388", "f389", "f390", "f391",
"f392", "f393", "f394", "f395", "f396", "f397", "f398", "f399", "f400",
"f401", "f402", "f403", "f404", "f405", "f406", "f407", "f408", "f409",
"f410", "f411", "f412", "f413", "f414", "f415", "f416", "f417", "f418",
"f419", "f420", "f421", "f422", "f423", "f424", "f425", "f426", "f427",
"f428", "f429", "f430", "f431", "f432", "f433", "f434", "f435", "f436",
"f437", "f438", "f439", "f440", "f441", "f442", "f443", "f444", "f445",
"f446", "f447", "f448", "f449", "f450", "f451", "f452", "f453", "f454",
"f455", "f456", "f457", "f458", "f459", "f460", "f461", "f462", "f463",
"f464", "f465", "f466", "f467", "f468", "f469", "f470", "f471", "f472",
"f473", "f474", "f475", "f476", "f477", "f478", "f479", "f480", "f481",
"f482", "f483", "f484", "f485", "f486", "f487", "f488", "f489", "f490",
"f491", "f492", "f493", "f494", "f495", "f496", "f497", "f498", "f499",
"f500", "f501", "f502", "f503", "f504", "f505", "f506", "f507", "f508",
"f509", "f510", "f511", "f512", "f513", "f514", "f515", "f516", "f517",
"f518", "f519", "f520", "f521", "f522", "f523", "f524", "f525", "f526",
"f527", "f528", "f529", "f530", "f531", "f532", "f533", "f534", "f535",
"f536", "f537", "f538", "f539", "f540", "f541", "f542", "f543", "f544",
"f545", "f546", "f547", "f548", "f549", "f550", "f551", "f552", "f553",
"f554", "f555", "f556", "f557", "f558", "f559", "f560", "f561", "f562",
"f563", "f564", "f565", "f566", "f567", "f568", "f569", "f570", "f571",
"f572", "f573", "f574", "f575", "f576", "f577", "f578", "f579", "f580",
"f581", "f582", "f583", "f584", "f585", "f586", "f587", "f588", "f589",
"f590", "f591", "f592", "f593", "f594", "f595", "f596", "f597", "f598",
"f599", "f600", "f601", "f602", "f603", "f604", "f605", "f606", "f607",
"f608", "f609", "f610", "f611", "f612", "f613", "f614", "f615", "f616",
"f617", "f618", "f619", "f620", "f621", "f622", "f623", "f624", "f625",
"f626", "f627", "f628", "f629", "f630", "f631", "f632", "f633", "f634",
"f635", "f636", "f637", "f638", "f639", "f640", "f641", "f642", "f643",
"f644", "f645", "f646", "f647", "f648", "f649", "f650", "f651", "f652",
"f653", "f654", "f655", "f656", "f657", "f658", "f659", "f660", "f661",
"f662", "f663", "f664", "f665", "f666", "f667", "f668", "f669", "f670",
"f671", "f672", "f673", "f674", "f675", "f676", "f677", "f678", "f679",
"f680", "f681", "f682", "f683", "f684", "f685", "f686", "f687", "f688",
"f689", "f690", "f691", "f692", "f693", "f694", "f695", "f696", "f697",
"f698", "f699", "f700", "f701", "f702", "f703", "f704", "f705", "f706",
"f707", "f708", "f709", "f710", "f711", "f712", "f713", "f714", "f715",
"f716", "f717", "f718", "f719", "f720", "f721", "f722", "f723", "f724",
"f725", "f726", "f727", "f728", "f729", "f730", "f731", "f732", "f733",
"f734", "f735", "f736", "f737", "f738", "f739", "f740", "f741", "f742",
"f743", "f744", "f745", "f746", "f747", "f748", "f749", "f750", "f751",
"f752", "f753", "f754", "f755", "f756", "f757", "f758", "f759", "f760",
"f761", "f762", "f763", "f764", "f765", "f766", "f767", "f768", "f769",
"f770", "f771", "f772", "f773", "f774", "f775", "f776", "f777", "f778",
"f779", "f780", "f781", "f782", "f783"],

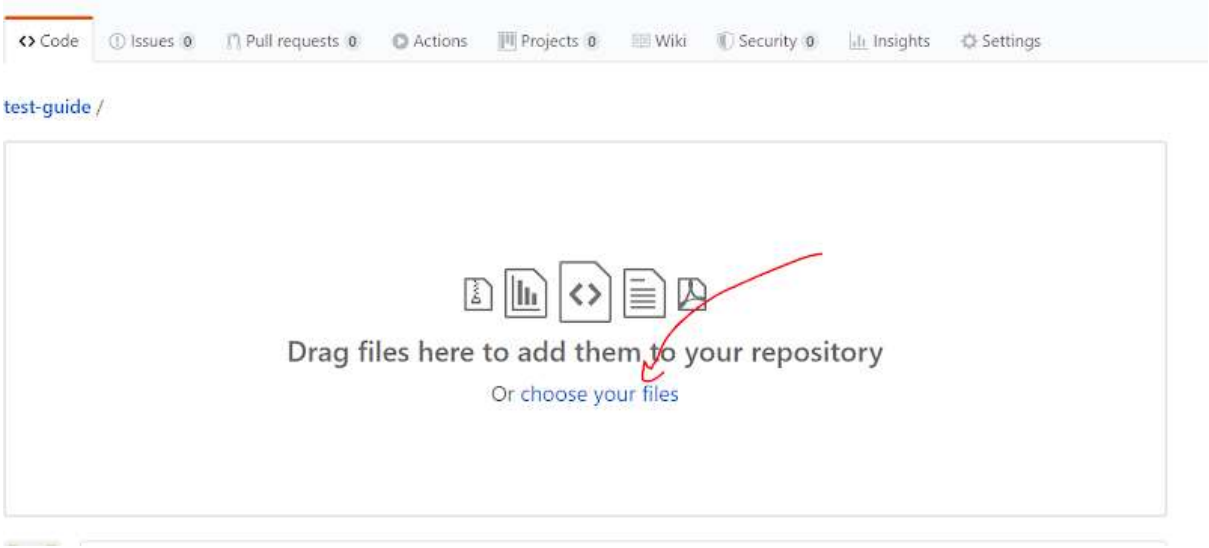
```
                        "Values": [ ]
                        }
                    },
                    "GlobalParameters": {}
        }

        p = JSON.parse("[" + dimgarray + "]");
        arg.Inputs.input1.Values.push(p);

        jQuery.ajax({
            url: "<your URL>",
            beforeSend: function (xhrObj) {
                xhrObj.setRequestHeader("Content-Type",
"application/json;charset=utf-8");
                xhrObj.setRequestHeader("Ocp-Apim-Subscription-Key", "<your
Subscription-Key>");

            },
            type: "POST",
            data: JSON.stringify(arg)
        })
            .done(function (data) {
                res = data.Results.output1.value.Values

                $.each(res, function (index, element) {
                        alert("Result: " + element)
                });
            })
            .fail(function () {
                alert("error");
            });
    }

    init();
}, false);
```

In line 146 paste your subscription key

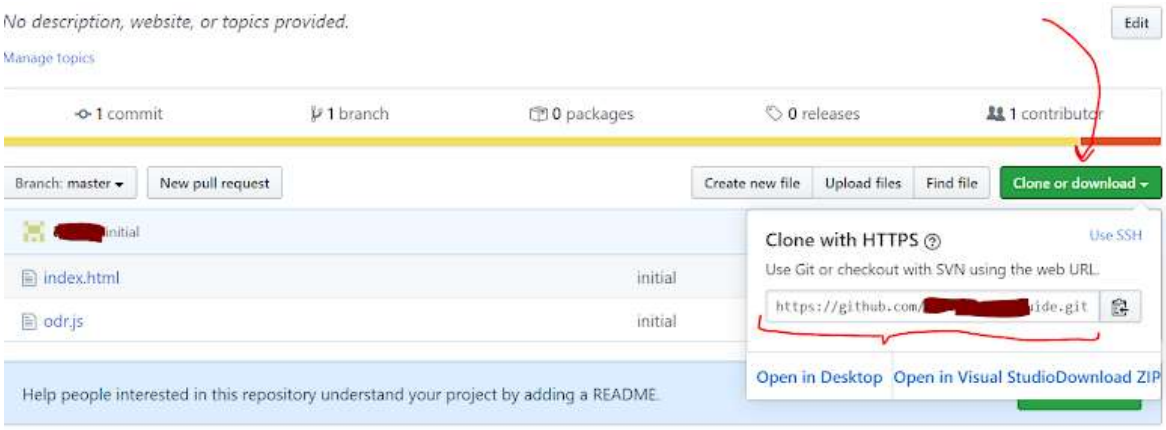


This javascript code will allow us to draw digits on a canvas and send the pixel data to our ML service to retrieve the prediction.

3. You can double click on the index.html file to open it from local copy and use it immediately. Instead we will publish it on Azure Web App service.
4. Create empty repository on your Github
5. Load index.html and odr.js and make commit



6. Go to https://shell.azure.com and make storage (check that you have activated student subscription)
7. Bash:
8. git clone <your Git url>



9. cd <your repository name>
10. az webapp up --location westeurope --name <your app_name> --html

```
psfedorov@Azure:~$ git clone https://github.com/psfed0r0v/ocrweb-test2.git
Cloning into 'ocrweb-test2'...
remote: Enumerating objects: 7, done.
remote: Counting objects: 100% (7/7), done.
remote: Compressing objects: 100% (5/5), done.
remote: Total 7 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (7/7), done.
Checking connectivity... done.
psfedorov@Azure:~$ cd ocrweb-test2
psfedorov@Azure:~/ocrweb-test2$ az webapp up --location westeurope --name <your app_name> --html
bash: your: No such file or directory
psfedorov@Azure:~/ocrweb-test2$
psfedorov@Azure:~/ocrweb-test2$ az webapp up --location westeurope --name ocrwebapp-test2 --html
Webapp ocrwebapp-test2 already exists. The command will deploy contents to the existing app.
psfedorov@Azure:~/ocrweb-test2$ az webapp up --location westeurope --name ocrwebapp-test4 --html
webapp ocrwebapp-test4 doesn't exist
Creating webapp 'ocrwebapp-test4' ...
Configuring default logging for the app, if not already enabled
Creating zip with contents of dir /home/psfedorov/ocrweb-test2 ...
Getting scm site credentials for zip deployment
Starting zip deployment. This operation can take a while to complete ...
Deployment endpoint responded with status code 202
You can launch the app at http://ocrwebapp-test4.azurewebsites.net
{
  "URL": "http://ocrwebapp-test4.azurewebsites.net",
  "appserviceplan": "psfedorov_asp_Windows_westeurope_0",
```

Fig. 33

11. navigate to http://<app_name>.azurewebsites.net

Test the solution

1. On the web page, draw any digit on the canvas with 28x28 grid. Press SEND button to call the web service with the pixel values drawn on the canvas. There will be a dialog box showing the

prediction result. (fig. 34)

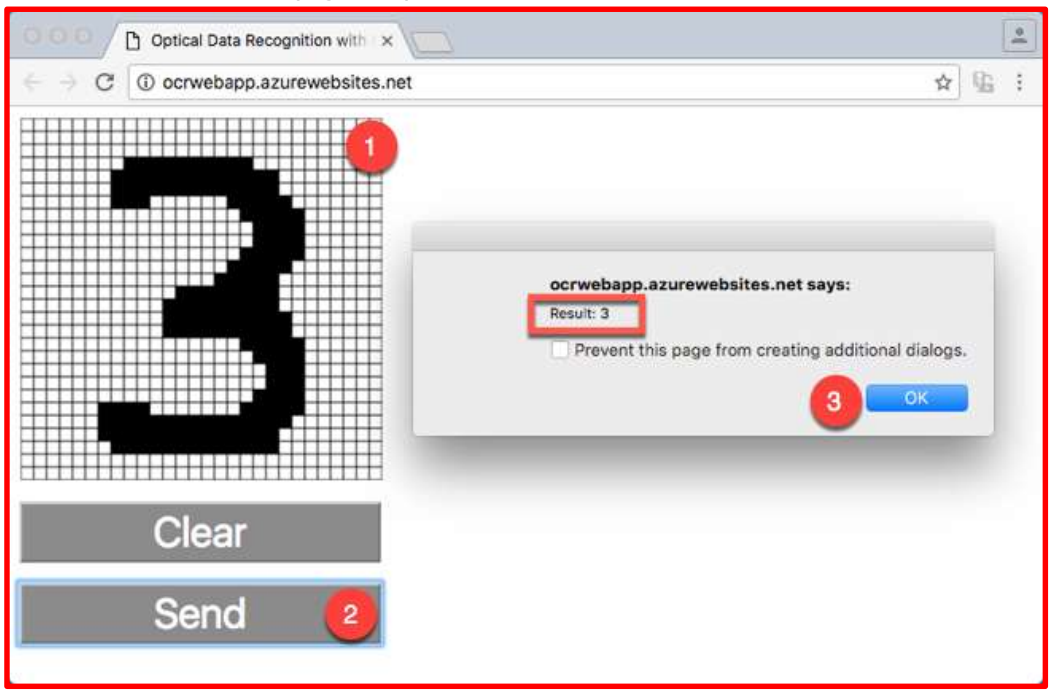


Fig. 34

There might be some error on the prediction. ML model may not recognize every of your drawing... You can try different classifier models and parameters in your ML model to improve the result. But moreover sending 784 features to recognize a digit is not efficient. We need to make feature engineering, find more meaningful and less features i.e. 5-10 features instead of 784 with more reliable prediction results!