

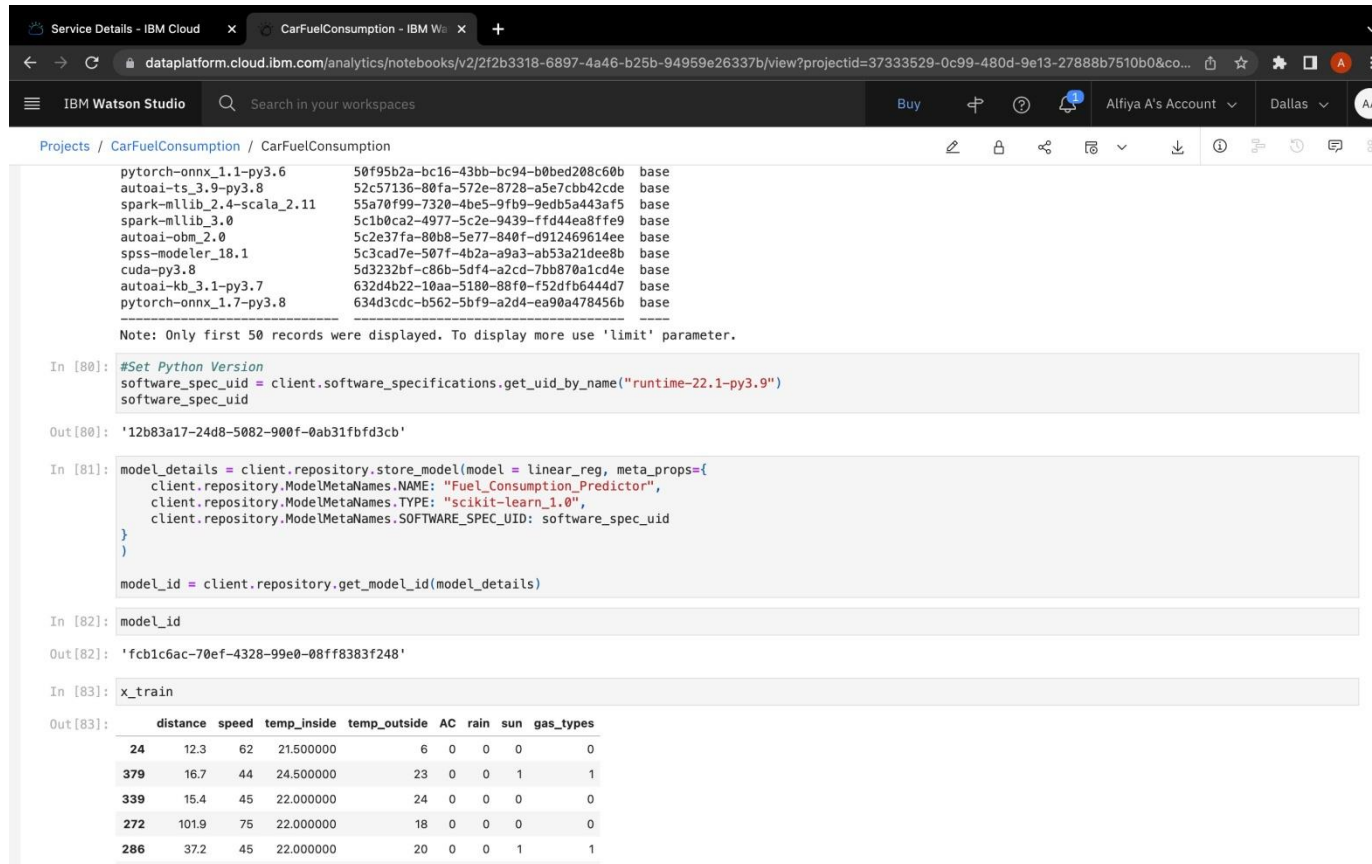
1.Setting up Watson Studio for running Jupyter notebook

The screenshot shows the IBM Watson Studio 'Assets' page. The top navigation bar includes 'Overview', 'Assets' (selected), 'Jobs', and 'Manage'. The main content area is titled 'All assets' and contains a table with two assets:

Name	Last modified
CarFuelConsumption Notebook	2 hours ago Modified by you
measurements2.xlsx XLSX	2 hours ago Modified by you

The left sidebar shows '2 assets' and 'All assets' selected. Under 'Asset types', there are 'Data' (1) and 'Notebooks' (1). The bottom of the page shows 'Items per page: 20' and '1-2 of 2 items'.

2. Training and saving the model in IBM Watson Machine Learning Service



The screenshot displays the IBM Watson Studio interface. The top navigation bar shows the project name 'CarFuelConsumption' and the user 'Alfiya A's Account'. The main workspace contains a Jupyter notebook with the following content:

```
pytorch-onnx_1.1-py3.6      50f95b2a-bc16-43bb-bc94-b0bed208c60b base
autoai-ts_3.9-py3.8         52c57136-80fa-572e-8728-a5e7cbb42cde base
spark-mllib_2.4-scala_2.11  55a70f99-7320-4be5-9fb9-9edb5a443af5 base
spark-mllib_3.0             5c1b0ca2-4977-5c2e-9439-ffd44ea8ffe9 base
autoai-obm_2.0              5c2e37fa-80b8-5e77-840f-d912469614ee base
spss-modeler_18.1           5c3cad7e-507f-4b2a-a9a3-ab53a21dee8b base
cuda-py3.8                  5d3232bf-c86b-5df4-a2cd-7bb870a1cd4e base
autoai-kb_3.1-py3.7         632d4b22-10aa-5180-88f0-f52dfb6444d7 base
pytorch-onnx_1.7-py3.8      634d3cdc-b562-5bf9-a2d4-ea90a478456b base
```

Note: Only first 50 records were displayed. To display more use 'limit' parameter.

```
In [80]: #Set Python Version
software_spec_uid = client.software_specifications.get_uid_by_name("runtime-22.1-py3.9")
software_spec_uid

Out[80]: '12b83a17-24d8-5082-900f-0ab31bfd3cb'
```

```
In [81]: model_details = client.repository.store_model(model = linear_reg, meta_props={
client.repository.ModelMetaNames.NAME: "Fuel_Consumption_Predictor",
client.repository.ModelMetaNames.TYPE: "scikit-learn_1.0",
client.repository.ModelMetaNames.SOFTWARE_SPEC_UID: software_spec_uid
})

model_id = client.repository.get_model_id(model_details)

In [82]: model_id

Out[82]: 'fcb1c6ac-70ef-4328-99e0-08ff8383f248'
```

```
In [83]: x_train
```

```
Out[83]:
```

	distance	speed	temp_inside	temp_outside	AC	rain	sun	gas_types
24	12.3	62	21.500000	6	0	0	0	0
379	16.7	44	24.500000	23	0	0	1	1
339	15.4	45	22.000000	24	0	0	0	0
272	101.9	75	22.000000	18	0	0	0	0
286	37.2	45	22.000000	20	0	0	1	1

3.Deployed the model in IBM Watson Machine Learning Service

Service Details - IBM Cloud

IBM Watson Studio

←

→

↺

dataplatfom.cloud.ibm.com/ml-runtime/spaces/85add7e-eec2-40e5-b535-55e72db5f0d5/deployments?context=cpdaas

🔗

☆

⚙

🖨

🔴 A

⋮

☰

IBM Watson Studio

🔍 Search in your workspaces

Buy

?

🔔

Alfiya A's Account

▼

Dallas

▼

AA

Deployments

/

📄

ⓘ

🔑

🕒

💬

⚙

models

Overview

Assets


Deployments

Jobs

Manage

🔍 Search

🔄

Name	Type	Status	Asset	Last modified	⌵	
 FuelConsumptionPredictor	Online	🟢 Deployed	Fuel_Consumption_Predictor	1 hour ago Alfiya A (You)		⋮

Items per page: 20

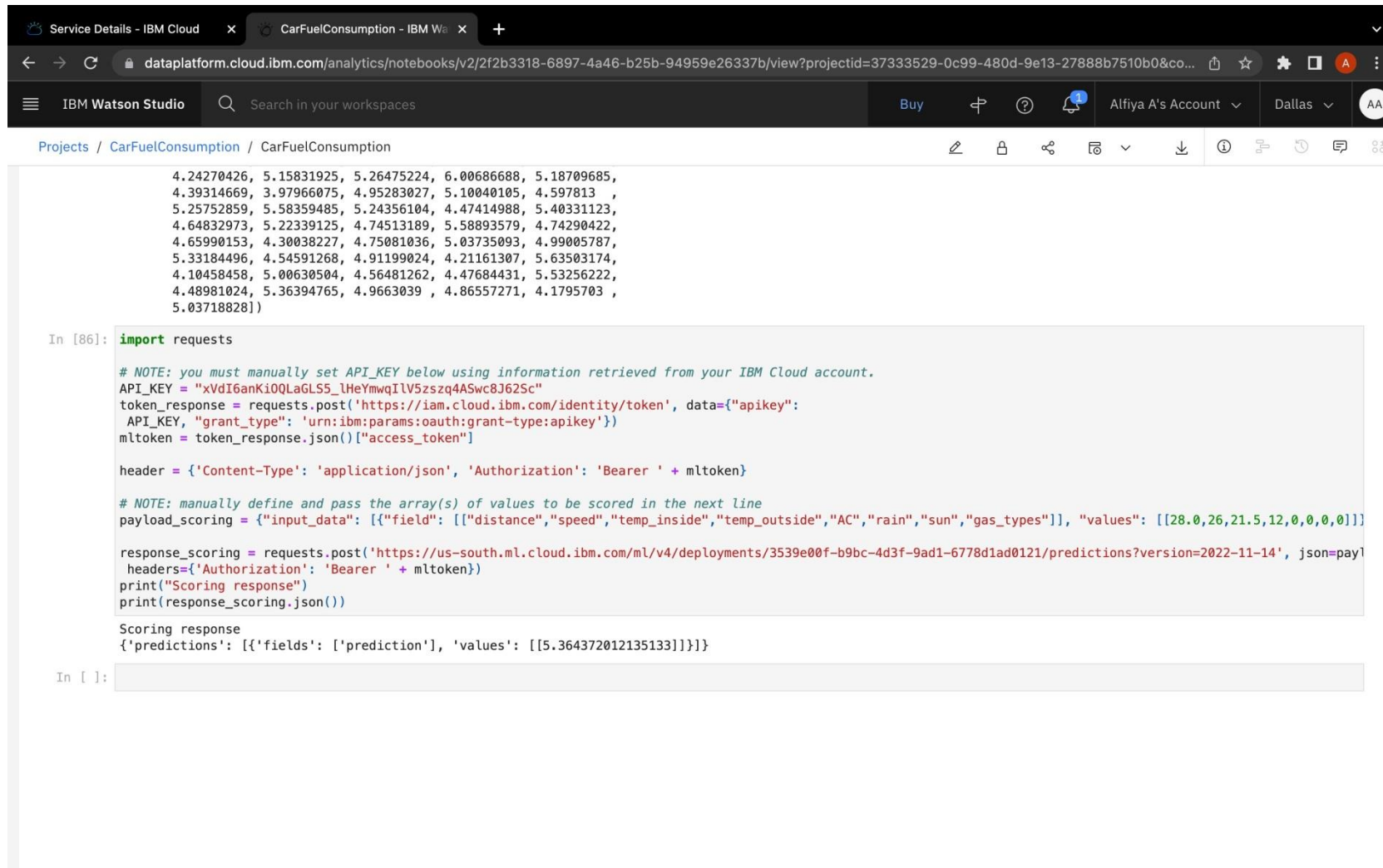
1-1 of 1 items

1 of 1 pages

◀

▶

4. Testing the created model using the API created for the deployed model



The screenshot displays the IBM Watson Studio web interface. The browser address bar shows the URL: `dataplatfrom.cloud.ibm.com/analytics/notebooks/v2/2f2b3318-6897-4a46-b25b-94959e26337b/view?projectId=37333529-0c99-480d-9e13-27888b7510b0&co...`. The interface includes a top navigation bar with 'Service Details - IBM Cloud' and 'CarFuelConsumption - IBM Wa' tabs. Below this is a search bar and user information for 'Alfiya A's Account' in the 'Dallas' region. The main content area shows a Jupyter notebook titled 'CarFuelConsumption'. The notebook contains a list of numerical data points and a code cell. The code cell, labeled 'In [86]:', imports the 'requests' library and includes comments about manually setting the API key. It then performs a POST request to the IBM Cloud IAM identity/token endpoint to obtain an access token. Subsequently, it constructs a header with the token and a payload for the 'input_data' field, which includes various sensor readings. Finally, it makes a POST request to the ML deployment endpoint for predictions, using the obtained token in the headers. The output of the code cell shows the 'Scoring response' as a JSON object containing a single prediction value.

```
4.24270426, 5.15831925, 5.26475224, 6.00686688, 5.18709685,
4.39314669, 3.97966075, 4.95283027, 5.10040105, 4.597813 ,
5.25752859, 5.58359485, 5.24356104, 4.47414988, 5.40331123,
4.64832973, 5.22339125, 4.74513189, 5.58893579, 4.74290422,
4.65990153, 4.30038227, 4.75081036, 5.03735093, 4.99005787,
5.33184496, 4.54591268, 4.91199024, 4.21161307, 5.63503174,
4.10458458, 5.00630504, 4.56481262, 4.47684431, 5.53256222,
4.48981024, 5.36394765, 4.9663039 , 4.86557271, 4.1795703 ,
5.03718828])

In [86]: import requests

# NOTE: you must manually set API_KEY below using information retrieved from your IBM Cloud account.
API_KEY = "xVdI6anKi0QLaGLS5_lHeYmwqILV5zsq4ASwc8J62Sc"
token_response = requests.post('https://iam.cloud.ibm.com/identity/token', data={"apikey":
API_KEY, "grant_type": 'urn:ibm:params:oauth:grant-type:apikey'})
mltoken = token_response.json()["access_token"]

header = {'Content-Type': 'application/json', 'Authorization': 'Bearer ' + mltoken}

# NOTE: manually define and pass the array(s) of values to be scored in the next line
payload_scoring = {"input_data": [{"field": ["distance","speed","temp_inside","temp_outside","AC","rain","sun","gas_types"]}, {"values": [[28.0,26,21.5,12,0,0,0,0]]}]

response_scoring = requests.post('https://us-south.ml.cloud.ibm.com/ml/v4/deployments/3539e00f-b9bc-4d3f-9ad1-6778d1ad0121/predictions?version=2022-11-14', json=payl
headers={'Authorization': 'Bearer ' + mltoken})
print("Scoring response")
print(response_scoring.json())

Scoring response
{'predictions': [{'fields': ['prediction'], 'values': [[5.364372012135133]]}]}
```

In []:

