

Lab 3 – ML OPs

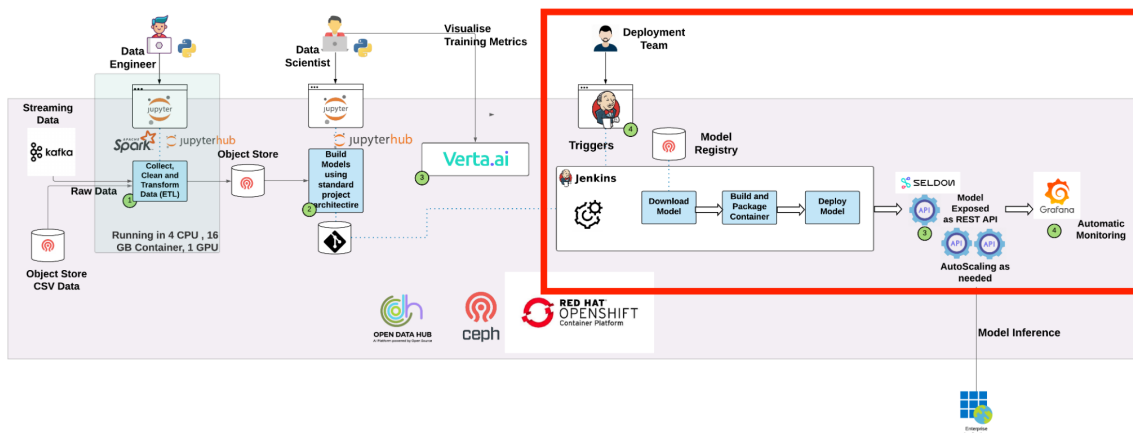
Introduction

Now as a data scientist, we've selected our chosen experiment, trained the model (a DecisionTreeClassifier) and finally we've pushed it to S3 Object storage.

Using the experiment id, as the identifier to retrieve from S3, we run our ML OPs pipeline and move the model and associated encoders to a *production* or *simulated production* environment. In this way, you add speed and potentially quality and security to this deployment process.

We also utilise the model serving component Seldon, which wraps the model behind a RESTful API, making the model easily available in this way for inference. As the flow is fully automated, we also ease and in fact eliminate the integration effort between application development and data scientists teams.

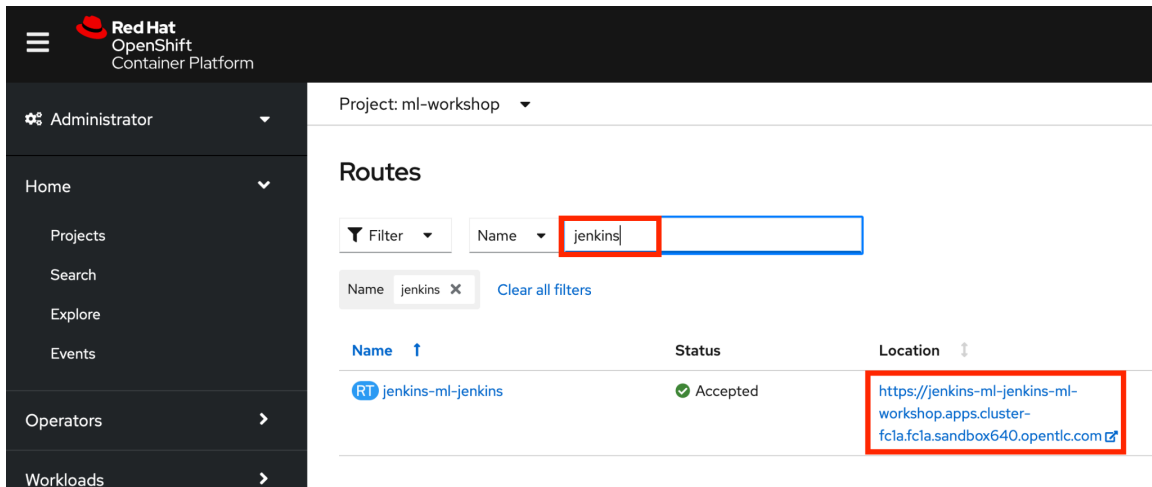
This diagram illustrates the workflow we're implementing - the ML OPs part of the overall AI/ML workflow:



Instructions to run the ML OPs workshop

Login to OpenShift using the credentials your administrator gave you. Ensure your workshop project ml-workshop is selected.

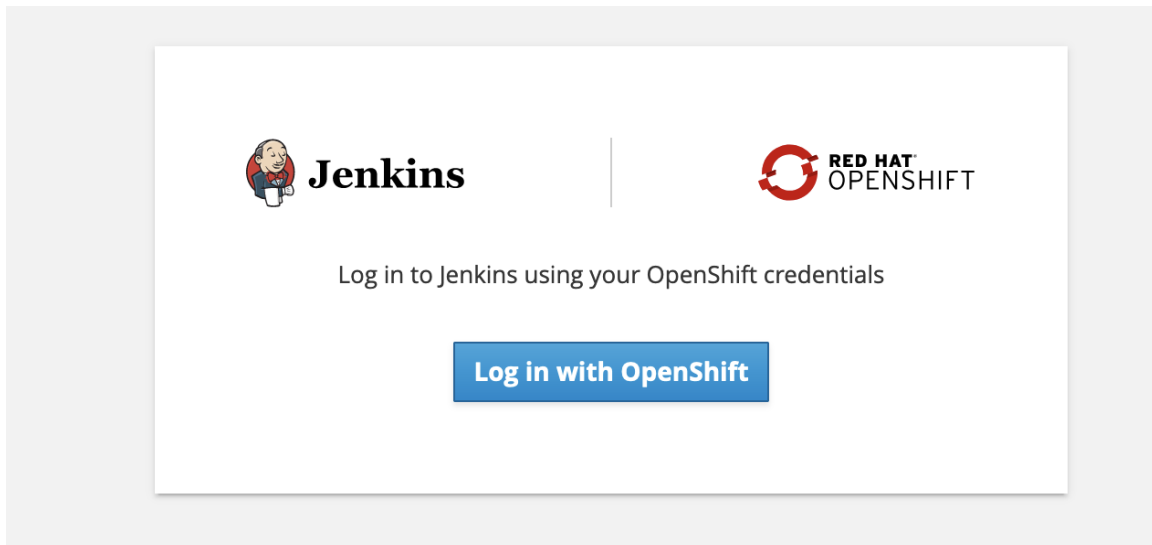
Choose the Administration dropdown , navigate to Network -> Routes. Filter on *jenkins* - and open the *Jenkins* route as shown.



The screenshot shows the OpenShift console interface. On the left is a sidebar with navigation options: Administrator, Home, Projects, Search, Explore, Events, Operators, and Workloads. The main area displays the 'Routes' page for the 'ml-workshop' project. A filter 'jenkins' is applied to the 'Name' column. A table lists the routes, with one entry 'jenkins-ml-jenkins' having a status of 'Accepted'. The 'Location' column for this route contains the URL 'https://jenkins-ml-jenkins-ml-workshop.apps.cluster-fcia.fcia.sandbox640.opentlc.com', which is highlighted with a red box.

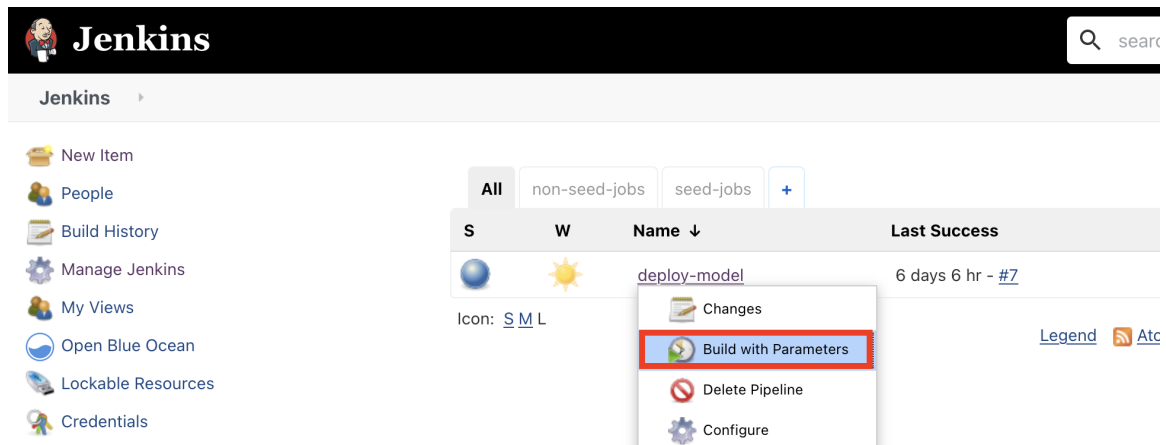
Name	Status	Location
jenkins-ml-jenkins	Accepted	https://jenkins-ml-jenkins-ml-workshop.apps.cluster-fcia.fcia.sandbox640.opentlc.com

When prompted enter your OpenShift credentials *userXX* and *openshift*, substituting *userXX* for your username.

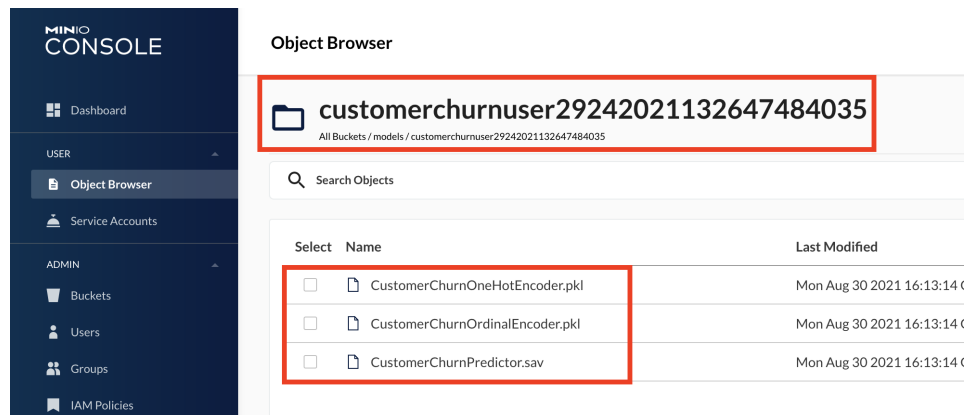


The screenshot shows the Jenkins login page. At the top, there are logos for Jenkins and Red Hat OpenShift. Below the logos, the text 'Log in to Jenkins using your OpenShift credentials' is displayed. A blue button with the text 'Log in with OpenShift' is centered at the bottom of the page.

Once in, you'll see a pipeline called `deploy-model`. Hover over it and an arrow will appear beside it. Choose **Build with Parameters** from the drop down menu.



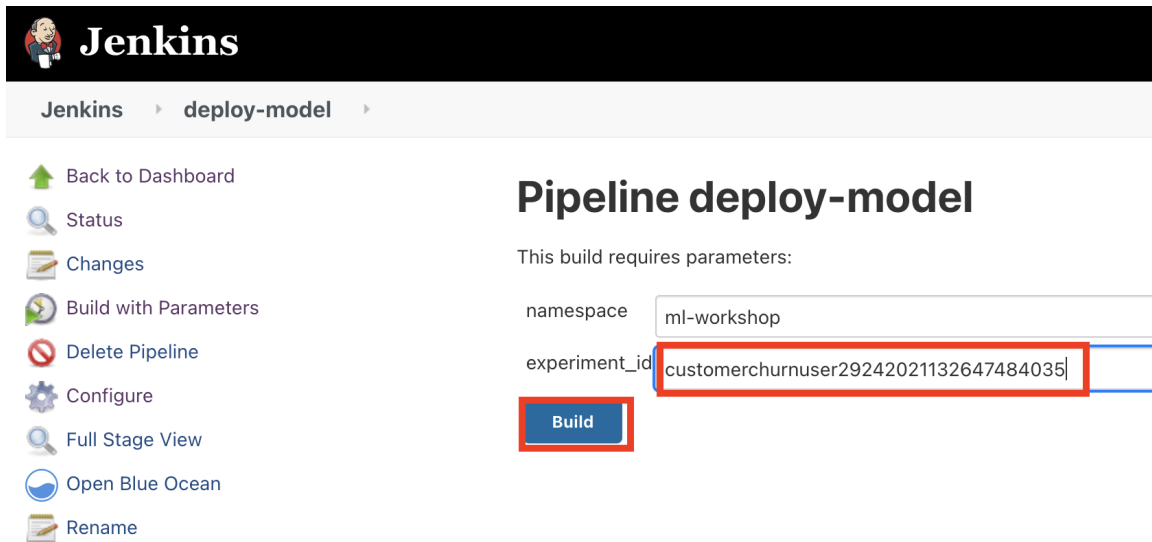
Remember the last exercise where you found your experiment outputs in the storage?



You need to use that experiment ID in this next step.

1. Enter the experiment id you retrieved at the end of the last workshop, in my case `customerchurnuser29242021132647484035`.
2. **Click Build**

AI/ML on OpenShift Workshop - Lab 4 - MLOps



Jenkins **deploy-model**

- Back to Dashboard
- Status
- Changes
- Build with Parameters
- Delete Pipeline
- Configure
- Full Stage View
- Open Blue Ocean
- Rename

Pipeline deploy-model

This build requires parameters:

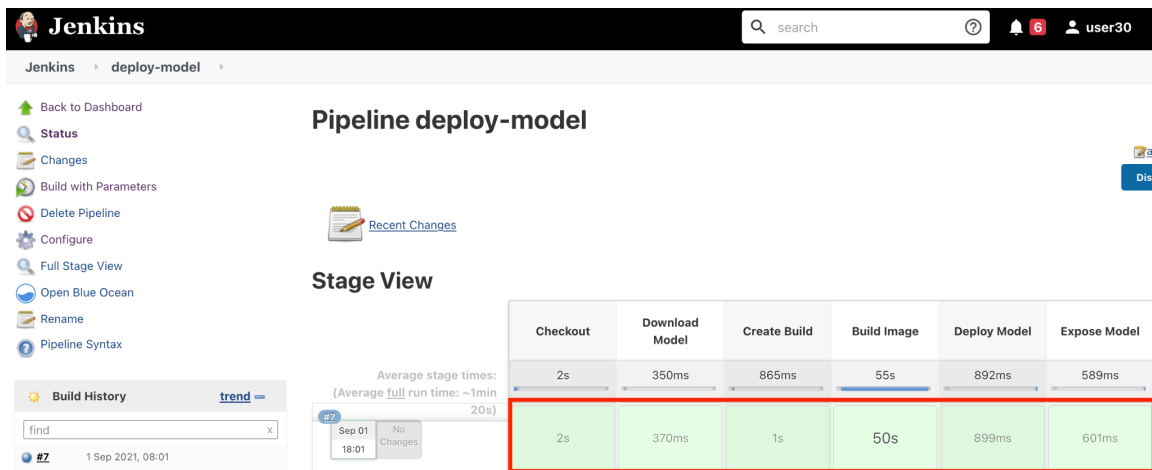
namespace

experiment_id

Build

Click **Build**

After a few seconds, you should see your pipeline begin and progress through the steps to completion you see highlighted here:



Jenkins **deploy-model**

Back to Dashboard

Status

Changes

Build with Parameters

Delete Pipeline

Configure

Full Stage View

Open Blue Ocean

Rename

Pipeline Syntax

Pipeline deploy-model

Recent Changes

Stage View

Average stage times:
(Average full run time: ~1min 20s)

Checkout	Download Model	Create Build	Build Image	Deploy Model	Expose Model
2s	350ms	865ms	55s	892ms	589ms
2s	370ms	1s	50s	899ms	601ms

Build History

find

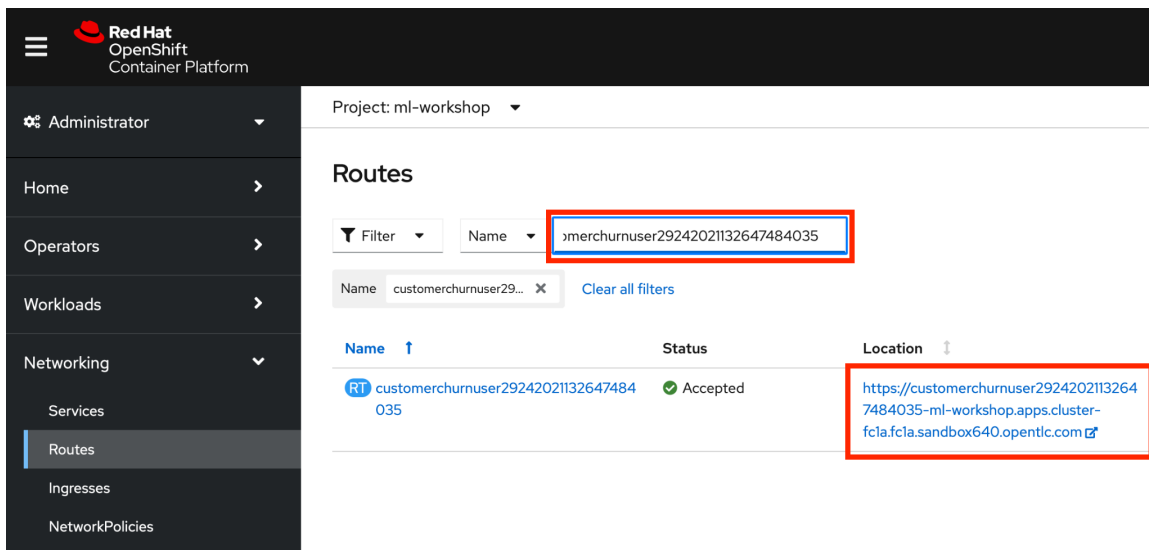
1 Sep 2021, 08:01

The pipeline then starts to package your model into an image and deploy it on to OpenShift. Proceed to the next section when the pipeline has completed.

Testing the Model via an API

You are now ready to test the API to your model.

1. Open the OpenShift tab on your browser.
2. Select the **Administrator** perspective in the left panel.
3. Click **Networking > Routes**
4. Filter on your experiment ID as shown below.



The screenshot shows the Red Hat OpenShift Container Platform Administrator interface. The left sidebar is set to the 'Administrator' perspective, and the 'Networking > Routes' section is selected. The main panel shows the 'Routes' page for the 'ml-workshop' project. A filter is applied to the 'Name' column with the value 'customerchurnuser29242021132647484035'. The table below shows a single route with the same name, status 'Accepted', and a location URL.

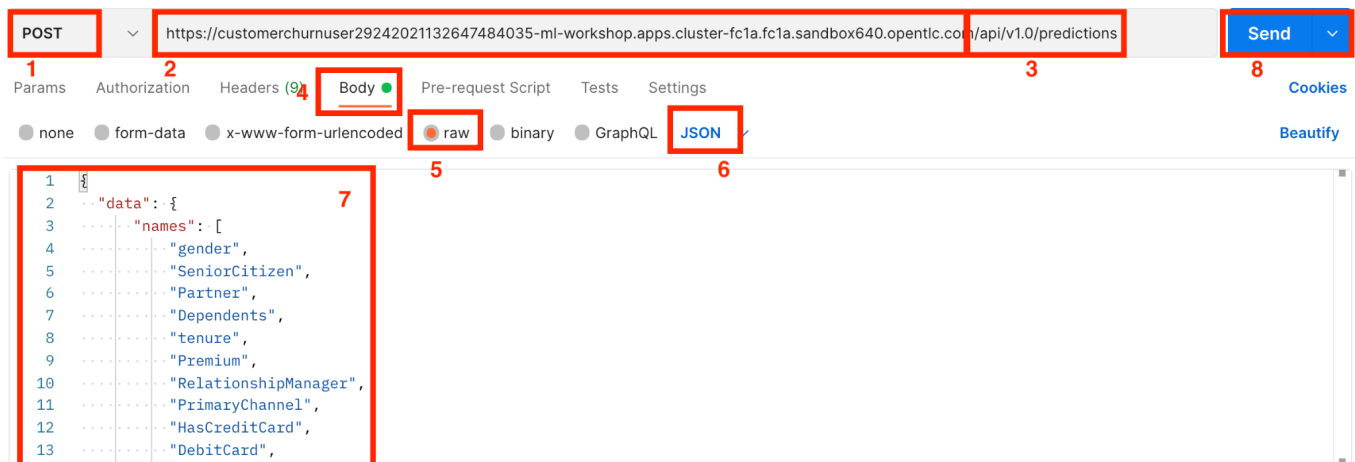
Name	Status	Location
customerchurnuser29242021132647484035	Accepted	https://customerchurnuser29242021132647484035-ml-workshop.apps.cluster-fc1a.fc1a.sandbox640.opentlc.com

The URL under the Locations column is the URL our pipeline has created for us - which we will use to make inference calls to our model. Make note of your URL, in my case

<https://customerchurnuser29242021132647484035-ml-workshop.apps.cluster-fc1a.fc1a.sandbox640.opentlc.com>

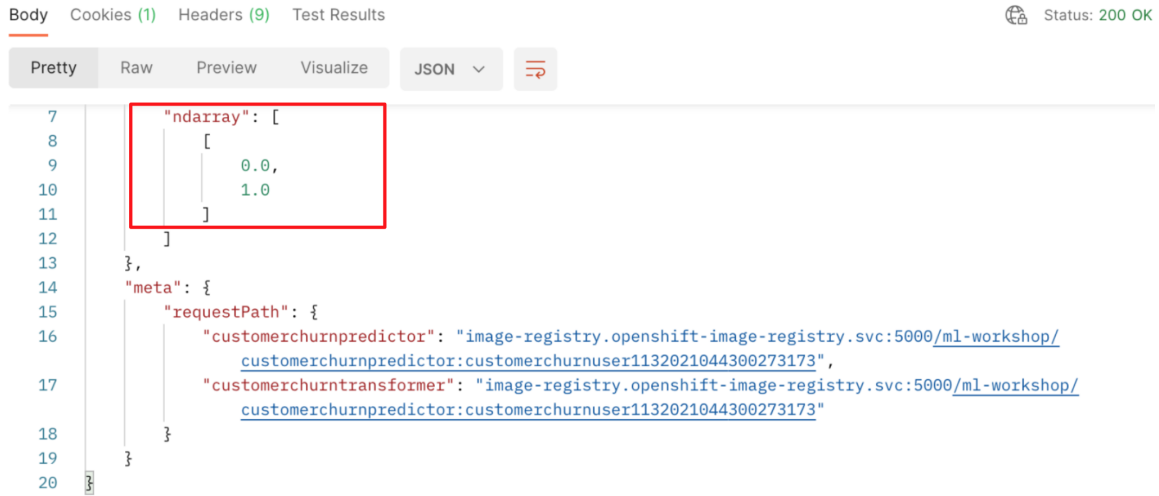
In order to make an inference call, you can use tools such as Postman, or the command line using curl, or there are various online options. I'll use *Postman* to illustrate. If you do not have Postman you can access a free web version here: <https://web.postman.co/home>

Open Postman, and create a new Workspace and add a new Request to it. Populate is as follows:



1. Choose the POST method
2. Take your inference URL from the previous step
3. append the path **/api/v1.0/predictions** to that to form the full URL, in my case
`https://customerchurnuser29242021132647484035-ml-workshop.apps.cluster-fc1a.fc1a.sandbox640.opentlc.com/api/v1.0/predictions`
4. Select Body
5. Choose raw
6. and JSON as shown as the content type.
7. Paste the JSON located below in [Appendix 1 - Sample Inference Request Body](#) into the Body box.
Notice this represents a customer, we're asking the model to predict how likely it is they will churn. Notice also, we're passing in string values such as Brokerage etc.
Conversion using the *Ordinal* and *One-Hot* encoders to numeric values will be done by the running container exposing the API.
This simplifies these API calls greatly for Application developers making these inference calls.
8. Click Send to make your inference call

Here you see a sample response.



```
Body Cookies (1) Headers (9) Test Results Status: 200 OK
Pretty Raw Preview Visualize JSON
7   "ndarray": [
8     [
9       0.0,
10      1.0
11    ]
12  ],
13  },
14  "meta": {
15    "requestPath": {
16      "customerchurnpredictor": "image-registry.openshift-image-registry.svc:5000/ml-workshop/
customerchurnpredictor:customerchurnuser1132021044300273173",
17      "customerchurntransformer": "image-registry.openshift-image-registry.svc:5000/ml-workshop/
customerchurnpredictor:customerchurnuser1132021044300273173"
18    }
19  }
20  }
```

Observe the True/False response. In this model we made a design choice to produce binary prediction: "Will likely Churn," and "Will not likely Churn" - which are represented as true (1.0) or false (0.0). An alternative approach could be to produce a likelihood as a percentage.

Appendix 1 - Sample Inference Request Body

```
{
  "data": {
    "names": [
      "gender",
      "SeniorCitizen",
      "Partner",
      "Dependents",
      "tenure",
      "Premium",
      "RelationshipManager",
      "PrimaryChannel",
      "HasCreditCard",
      "DebitCard",
      "IncomeProtection",
      "WealthManagement",
      "HomeEquityLoans",
      "MoneyMarketAccount",
      "CreditRating",
      "PaperlessBilling",
      "AccountType",
      "MonthlyCharges",
      "TotalCharges"
    ],
    "ndarray": [
      [
        "Female",
        1,
        "Yes",
        "No",
        0,
        "Yes",
        "Yes",
        "Mobile",
        "Yes",
        "Yes",
        "No",
        "No",
        "Yes",
        "Yes",
        "Low",
        "No",
        "Brokerage",
        100,
        300
      ]
    ]
  }
}
```