

Cloud Application Development

Phase 4: Project Development Part 2

Project title:

Machine learning model deployment with IBM cloud Watson Studio.

Problem Statement:

Become a wizard of predictive analytics with IBM Cloud Watson Studio. Train machine learning models to predict the outcomes in real time. Deploy the models as web services and integrate them into your applications. Unlock the magic of data driven insights and make informed decisions like never before.

Contents of the document:

- Installing the required libraries.
- Adding Watson Machine Learning service credentials to access.
- WatsonMachineLearningAPIClient library.
- Storing the model in WML repository.

Step 1: Installing the Required Libraries

In a house price prediction project, we'll typically need libraries for data manipulation, visualization, and machine learning.

Here's a list of some commonly used libraries in Python along with their installation instructions:

1. **NumPy** (Numerical Python):

- Installation: `pip install numpy`

2. **Pandas** (Python Data Analysis Library):

- Installation: `pip install pandas`

3. **Matplotlib** (Data Visualization):

- Installation: `pip install matplotlib`

4. **Seaborn** (Statistical Data Visualization):

- Installation: `pip install seaborn`

5. **Scikit-Learn** (Machine Learning):

- Installation: `pip install scikit-learn`

6. **LightGBM** (Gradient Boosting Library):

- Installation: `pip install lightgbm`

7. **TensorFlow** or **PyTorch** (Deep Learning, if needed):

- TensorFlow Installation: `pip install tensorflow`
- PyTorch Installation: Visit PyTorch website for installation instructions

8. **Jupyter Notebook** (Interactive Development Environment):

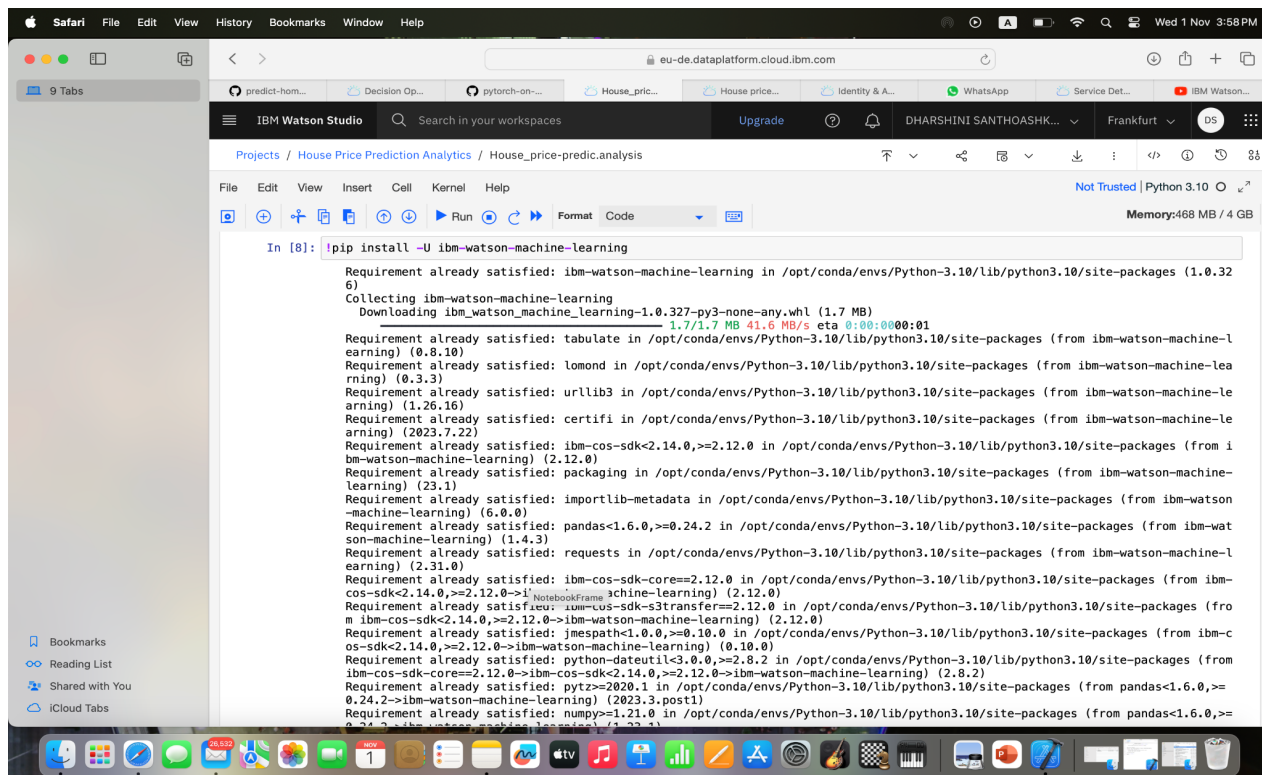
- Installation: `pip install jupyter`

9. **Scipy** (Scientific Computing):

- Installation: pip install scipy

10. Statsmodels (Statistical Modeling):

- Installation: pip install statsmodels



The screenshot shows the IBM Watson Studio web interface in a Safari browser. The top navigation bar includes the IBM Watson Studio logo, a search bar, and user information. The main workspace area displays a Jupyter Notebook with a terminal output showing the installation of the 'ibm-watson-machine-learning' package. The output indicates that the package is already installed in the current environment, and it lists various dependencies that are also satisfied. The interface includes a file explorer on the left and a top toolbar with various icons for file management and execution.

```
In [8]: pip install -U ibm-watson-machine-learning

Requirement already satisfied: ibm-watson-machine-learning in /opt/conda/envs/Python-3.10/lib/python3.10/site-packages (1.0.326)
Collecting ibm-watson-machine-learning
  Downloading ibm_watson_machine_learning-1.0.327-py3-none-any.whl (1.7 MB)
    1.7/1.7 MB 41.6 MB/s eta 0:00:00:01
Requirement already satisfied: tabulate in /opt/conda/envs/Python-3.10/lib/python3.10/site-packages (from ibm-watson-machine-learning) (0.8.10)
Requirement already satisfied: lomond in /opt/conda/envs/Python-3.10/lib/python3.10/site-packages (from ibm-watson-machine-learning) (0.3.3)
Requirement already satisfied: urllib3 in /opt/conda/envs/Python-3.10/lib/python3.10/site-packages (from ibm-watson-machine-learning) (1.26.16)
Requirement already satisfied: certifi in /opt/conda/envs/Python-3.10/lib/python3.10/site-packages (from ibm-watson-machine-learning) (2023.7.22)
Requirement already satisfied: ibm-cos-sdk<2.14.0,>=2.12.0 in /opt/conda/envs/Python-3.10/lib/python3.10/site-packages (from ibm-watson-machine-learning) (2.12.0)
Requirement already satisfied: packaging in /opt/conda/envs/Python-3.10/lib/python3.10/site-packages (from ibm-watson-machine-learning) (23.1)
Requirement already satisfied: importlib-metadata in /opt/conda/envs/Python-3.10/lib/python3.10/site-packages (from ibm-watson-machine-learning) (6.0.0)
Requirement already satisfied: pandas<1.6.0,>=0.24.2 in /opt/conda/envs/Python-3.10/lib/python3.10/site-packages (from ibm-watson-machine-learning) (1.4.3)
Requirement already satisfied: requests in /opt/conda/envs/Python-3.10/lib/python3.10/site-packages (from ibm-watson-machine-learning) (2.31.0)
Requirement already satisfied: ibm-cos-sdk-core==2.12.0 in /opt/conda/envs/Python-3.10/lib/python3.10/site-packages (from ibm-cos-sdk<2.14.0,>=2.12.0->ibm-watson-machine-learning) (2.12.0)
Requirement already satisfied: ibm-cos-sdk-s3transfer==2.12.0 in /opt/conda/envs/Python-3.10/lib/python3.10/site-packages (from ibm-cos-sdk<2.14.0,>=2.12.0->ibm-watson-machine-learning) (2.12.0)
Requirement already satisfied: jmespath<1.0.0,>=0.10.0 in /opt/conda/envs/Python-3.10/lib/python3.10/site-packages (from ibm-cos-sdk<2.14.0,>=2.12.0->ibm-watson-machine-learning) (0.10.0)
Requirement already satisfied: python-dateutil<3.0.0,>=2.8.2 in /opt/conda/envs/Python-3.10/lib/python3.10/site-packages (from ibm-cos-sdk-core==2.12.0->ibm-cos-sdk<2.14.0,>=2.12.0->ibm-watson-machine-learning) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /opt/conda/envs/Python-3.10/lib/python3.10/site-packages (from pandas<1.6.0,>=0.24.2->ibm-watson-machine-learning) (2023.3.post1)
Requirement already satisfied: numpy>=1.21.0 in /opt/conda/envs/Python-3.10/lib/python3.10/site-packages (from pandas<1.6.0,>=0.24.2->ibm-watson-machine-learning) (1.24.3)
```

Step
2:

Adding Watson Machine Learning service credentials to access

To access IBM Watson Machine Learning (WML) service, you will need the service credentials, typically provided when you set up your Watson Machine Learning instance on IBM Cloud. Here's how you can add these credentials to your project:

- Obtain Watson Machine Learning Service Credentials:

First, you need to obtain the service credentials for your Watson Machine Learning instance. You can usually find these credentials in the IBM Cloud Console or by following the steps provided in your Watson

Machine Learning instance documentation. Your credentials should include information like the API key, instance ID, and URL.

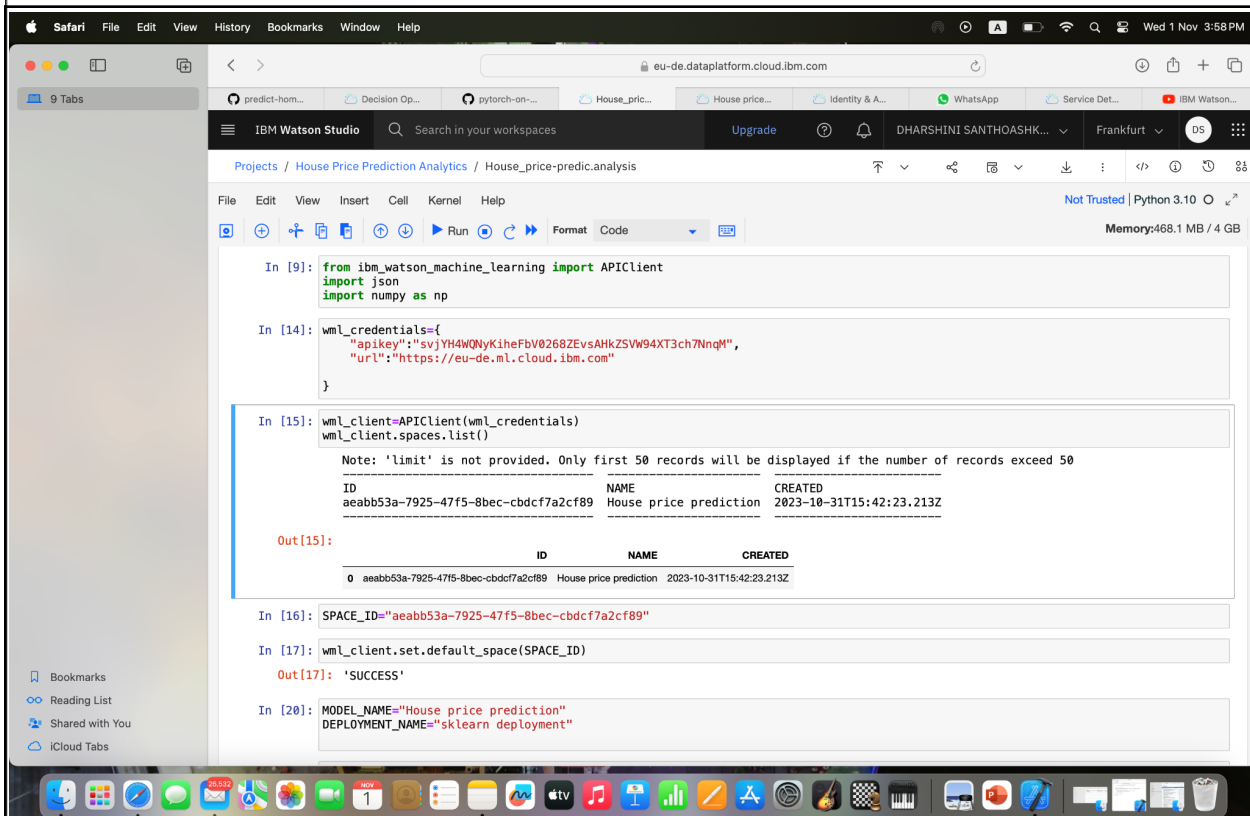
- Store Credentials Securely:

It's crucial to store your credentials securely. You can use environment variables or configuration files for this purpose. Never hardcode credentials directly into your code to keep them safe.

- Use the API Client:

With the `APIClient`, you can interact with your Watson Machine Learning instance to deploy and manage machine learning models, create deployments, and more.

By following these steps, we can securely add our Watson Machine Learning service credentials to our project.



The screenshot shows the IBM Watson Studio interface in a Safari browser. The main workspace displays a Jupyter Notebook with the following code and output:

```
In [9]: from ibm_watson_machine_learning import APIClient
import json
import numpy as np

In [14]: wml_credentials={
    "apikey": "svjYH4WQnyKiheFbV0268ZEvSAHkZSVW94XT3ch7NnqM",
    "url": "https://eu-de.ml.cloud.ibm.com"
}

In [15]: wml_client=APIClient(wml_credentials)
wml_client.spaces.list()

Note: 'limit' is not provided. Only first 50 records will be displayed if the number of records exceed 50
```

ID	NAME	CREATED
aeabb53a-7925-47f5-8bec-cbdcf7a2cf89	House price prediction	2023-10-31T15:42:23.213Z

```
Out[15]:
```

ID	NAME	CREATED
0 aeabb53a-7925-47f5-8bec-cbdcf7a2cf89	House price prediction	2023-10-31T15:42:23.213Z

```
In [16]: SPACE_ID="aeabb53a-7925-47f5-8bec-cbdcf7a2cf89"

In [17]: wml_client.set.default_space(SPACE_ID)

Out[17]: 'SUCCESS'

In [20]: MODEL_NAME="House price prediction"
DEPLOYMENT_NAME="sklearn deployment"
```

Step 3: WatsonMachineLearningAPIClient library

To create a Watson Machine Learning API client library for a house price prediction project, we'll need to follow these general steps:

1. Set up your environment:

- Ensure you have access to the IBM Watson Machine Learning service.
- Obtain API credentials and an API key for your Watson Machine Learning service.

2. Select a library or framework:

- For Python, you can use libraries like requests or the ibm-watson SDK.

3. Define API endpoints:

- Define the relevant API endpoints for your house price prediction project. These may include endpoints for:
 - Deploying machine learning models
 - Making predictions
 - Managing deployments

4. Implement authentication:

- Use your API credentials and key to implement authentication in your client library. This usually involves adding the necessary headers or tokens to your HTTP requests.

5. Create client functions:

- Define functions in your client library to perform specific actions related to your project, such as:
 - Deploying a machine learning model

- Making predictions based on input data
- Managing model deployments (e.g., starting, stopping, deleting)

6. Implement data preprocessing:

- Consider adding data preprocessing functions to prepare input data for making predictions with your machine learning model.

7. Testing:

- Create unit tests for your client library to ensure that it functions as expected.

8. Usage examples:

- Provide example code and use cases for the client library to help users understand how to integrate it into their projects.

9. Publish and share:

- Share the client library on relevant platforms, such as GitHub, to make it accessible to others.

The screenshot displays the IBM Watson Studio web interface in a Safari browser. The active workspace is 'House Price Prediction Analytics'. The Jupyter notebook shows the following code and output:

```

Out[15]:
      ID                NAME                CREATED
0 aeabb53a-7925-47f5-8bec-cbdcf7a2cf89 House price prediction 2023-10-31T15:42:23.213Z

In [16]: SPACE_ID="aeabb53a-7925-47f5-8bec-cbdcf7a2cf89"

In [17]: wml_client.set_default_space(SPACE_ID)
Out[17]: 'SUCCESS'

In [20]: MODEL_NAME="House price prediction"
DEPLOYMENT_NAME="sklearn deployment"

In [21]: software_spec_uid=wml_client.software_specifications.get_id_by_name('default_py3.7')

In [22]: model_props={
wml_client.repository.ModelMetaNames.NAME:MODEL_NAME,
wml_client.repository.ModelMetaNames.TYPE:'scikit-learn_0.23',
wml_client.repository.ModelMetaNames.NAME:MODEL_NAME,
}

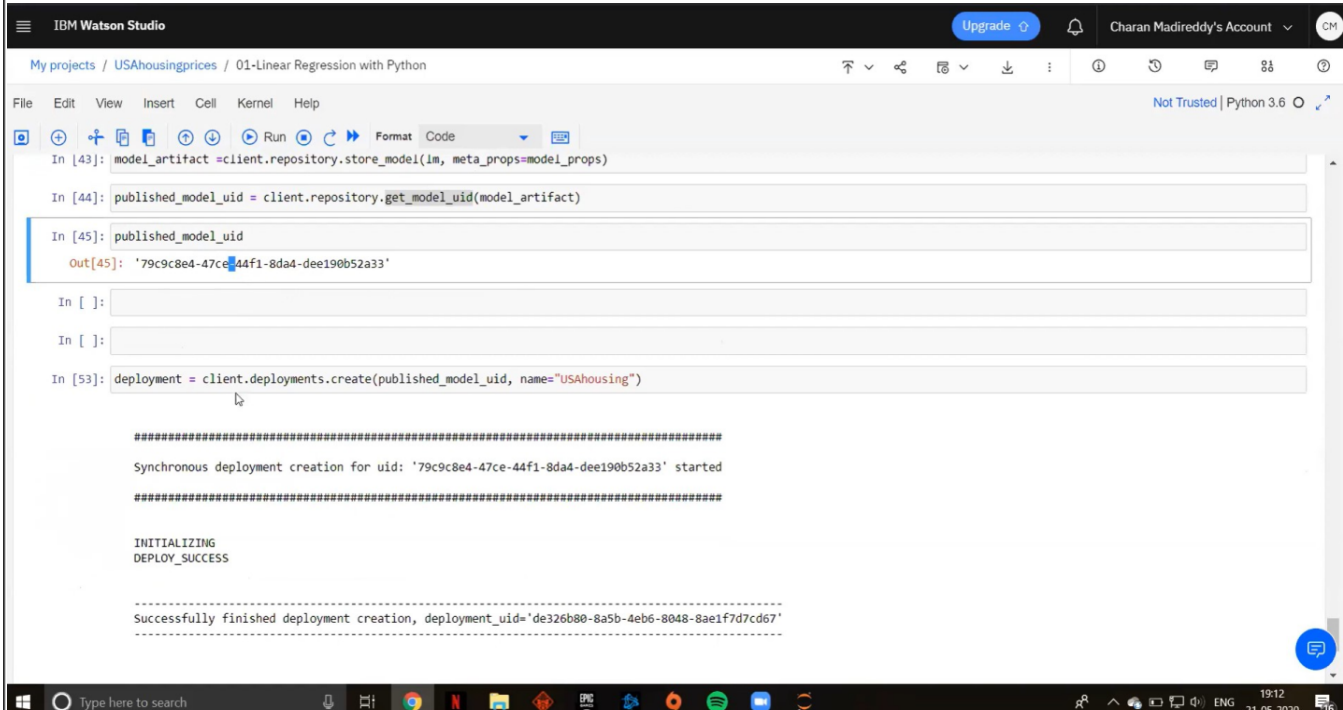
In [23]: software_spec_uid="e4429883-c883-42b6-87a8-f419d64088cd"

In [24]: model_details = wml_client.repository.store_model(
model=classifier,
meta_props=model_props,
training_data=X_train.head(),
training_target=y_train.head()
)
  
```

The interface includes a top navigation bar with 'Upgrade' and user information, a left sidebar with '9 Tabs', and a bottom dock with various system icons. The notebook's status bar indicates 'Not Trusted', 'Python 3.10', and 'Memory: 468.1 MB / 4 GB'.

Step 4: Storing the model in WML repository

Successfully we have deployed our model in IBM Watson Studio.



The screenshot displays the IBM Watson Studio web interface. At the top, the header shows 'IBM Watson Studio' with an 'Upgrade' button and a user profile for 'Charan Madireddy's Account'. The breadcrumb navigation indicates the current location: 'My projects / USAhousingprices / 01-Linear Regression with Python'. Below the header is a standard Jupyter Notebook toolbar with options like 'File', 'Edit', 'View', 'Insert', 'Cell', 'Kernel', and 'Help'. The main area contains a series of code cells. The first cell (In [43]) stores the model artifact. The second cell (In [44]) retrieves the published model UID. The third cell (In [45]) displays the UID as an output. The fourth cell (In [53]) creates a deployment named 'USAhousing'. The output of the final cell shows a status message: 'Synchronous deployment creation for uid: '79c9c8e4-47ce-44f1-8da4-dee190b52a33' started', followed by 'INITIALIZING' and 'DEPLOY_SUCCESS', and finally 'Successfully finished deployment creation, deployment_uid='de326b80-8a5b-4eb6-8048-8ae1f7d7cd67''.

```
IBM Watson Studio Upgrade Charan Madireddy's Account CM

My projects / USAhousingprices / 01-Linear Regression with Python

File Edit View Insert Cell Kernel Help Not Trusted | Python 3.6

In [43]: model_artifact = client.repository.store_model(lm, meta_props=model_props)

In [44]: published_model_uid = client.repository.get_model_uid(model_artifact)

In [45]: published_model_uid
Out[45]: '79c9c8e4-47ce-44f1-8da4-dee190b52a33'

In [ ]:

In [ ]:

In [53]: deployment = client.deployments.create(published_model_uid, name="USAhousing")

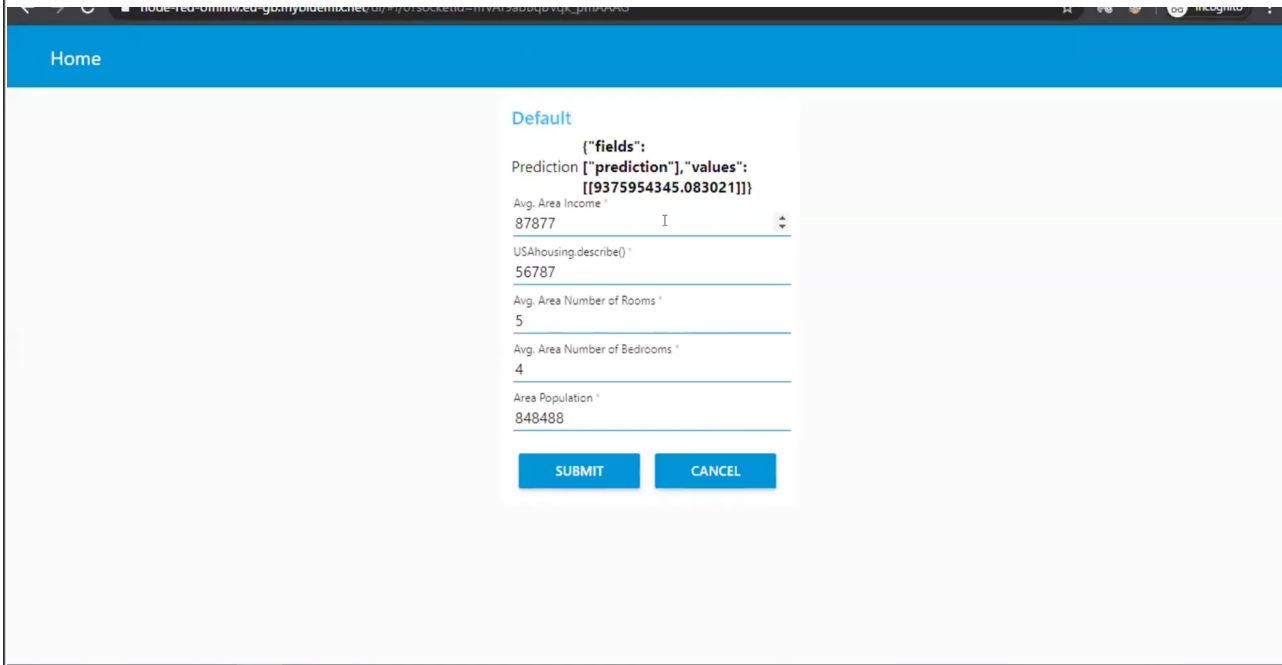
#####
Synchronous deployment creation for uid: '79c9c8e4-47ce-44f1-8da4-dee190b52a33' started
#####

INITIALIZING
DEPLOY_SUCCESS

.....
Successfully finished deployment creation, deployment_uid='de326b80-8a5b-4eb6-8048-8ae1f7d7cd67'
.....
```

Step 5:

Now we have deployed our machine learning model as a Web service. Once the model is deployed ,it can be used to make predictions or provide other intelligent services to web users.



The screenshot shows a web application interface with a blue header bar labeled "Home". Below the header, there is a "Default" section. This section displays a JSON-like prediction output: `{"fields": [{"prediction": [{"values": [{"9375954345.083021}]}]}`. Below this, several input fields are shown with their corresponding values: "Avg. Area Income" (87877), "USAhousing.describe()" (56787), "Avg. Area Number of Rooms" (5), "Avg. Area Number of Bedrooms" (4), and "Area Population" (848488). At the bottom of the form, there are two buttons: "SUBMIT" and "CANCEL".

Field	Value
Avg. Area Income *	87877
USAhousing.describe() *	56787
Avg. Area Number of Rooms *	5
Avg. Area Number of Bedrooms *	4
Area Population *	848488