

Cloud Application Development

Phase 5: Project Documentation & Submission

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 outcomes in real-time. Deploy the models as web services and integrate them into your applications.

Problem definition:

The project involves training a machine learning model using IBM Cloud Watson Studio and deploying it as a web service. The goal is to become proficient in predictive analytics by creating a model that can predict outcomes in real-time. The project encompasses defining the predictive use case, selecting a suitable dataset, training a machine learning model, deploying the model as a web service, and integrating it into applications.

Project objective:

A cloud-based predictive analytics solution has been designed using IBM Cloud Watson Studio that empowers users to train, deploy, and integrate machine learning models into their applications. The goal is to harness the power of data-driven insights, enabling users to make informed decisions in real-time and unlocking the magic of predictive analytics.

Design thinking Process:

To address the problem statement of becoming a wizard of predictive analytics with IBM Cloud Watson Studio, you can follow a design thinking process with the following steps:

Empathize:

Understand the target users or audience who want to become wizards of predictive analytics.

Define:

Clearly define the problem you're solving, considering the users' perspective.

Ideate:

Brainstorm creative solutions to address the defined problem.

Prototype:

Create a prototype or mockup of the cloud application that will help users become predictive analytics wizards.

Test:

Gather feedback from potential users by allowing them to interact with the prototype.

Implement:

Develop the cloud application based on the refined prototype.

Deploy:

Deploy the application on a cloud platform, such as IBM Cloud, for accessibility and scalability.

Iterate:

Continuously gather user feedback and make improvements to the application.

Predictive use case :**Description:**

PMML is an XML based language used to represent predictive models created as the result of the predictive modeling process. It allows for predictive models to be easily shared between applications.

Key Components:

- Data Collection
- Data Preprocessing

- Model Selection
- Model Training
- Validation and Evaluation
- Fine-tuning

Dataselection:

Data collection for a dataset with labels is a critical step in building a machine learning model for tasks like classification or prediction.

The "House Price Prediction" dataset contains information about residential properties and their sale prices. It's used to build predictive models for estimating house prices, with features like bedrooms, bathrooms, and square footage. Data scientists use it for regression practice, and it has applications in real estate, finance, and urban planning. Various versions of this dataset may focus on specific regions or property types.

Model Training:

The decision tree algorithm is a machine learning method used for "House Price Prediction" dataset in Scikit-Learn (sklearn). It works by recursively splitting data based on features to create a tree structure that predicts house prices. Scikit-Learn's implementation allows users to easily build, train, and evaluate decision tree models for predicting house prices from dataset features.

Model Deployment and Integration:

Deploying a trained model as a web service on IBM Cloud Watson Studio typically involves the following steps:

1. Prepare the Model
2. Create a Watson Machine Learning (WML) Instance
3. Package Dependencies
4. Deployment Configuration
5. Deploy the Model
6. Endpoint Testing

The deployed model has to be integrated into web applications or other systems that need to use the object detection service. We can make HTTP requests to the endpoint

to get predictions for new images.

DEVELOPMENT PHASE :

STEP 1:Set Up IBM Watson Studio

1. Create an IBM Cloud Account:

If we don't already have an IBM Cloud account, we'll need to create one. Visit the IBM Cloud website (<https://cloud.ibm.com>) and sign up for a free account

2. Access IBM Watson Studio: After creating our IBM Cloud account, log in to IBM Cloud

3. Create a New Project

i. Once logged in, navigate to the IBM Watson Studio service.

ii. Create a new project within Watson Studio to organize your work. Give your project a name and, if applicable, a description.

4. Define the Project Type

5. Configure the Project:

6. Invite Collaborators

7. Data Import

Step 2:Data Collection and Preparation:

Data Collection:

- **Identify Data Sources:** Determine where our data will come from. It can be internal databases, external sources, or a combination of both. Identify the specific data needed to address your predictive analytics problem.
- **Data Retrieval:** Obtain the data from the identified sources. Depending on the source, we may need to use SQL queries, web scraping, APIs, or other methods to retrieve the data.
- **Data Import:** Import the collected data into your IBM Watson Studio project. We can typically do this via the project interface or using programming languages like Python or R.

Data Preparation:

Data Inspection:

- o Explore the imported data to understand its structure. Use basic commands and tools to check the data's dimensions, data types, and an initial sample of records.

Data Cleaning:

- o Address missing data: Identify and handle missing values using techniques like imputation, removal, or interpolation.
- o Remove duplicates: Check for and eliminate duplicate records from the dataset.
- o Outlier detection and handling: Identify outliers and decide whether to remove, transform, or keep them based on their impact on the analysis.
- o Data normalization: If your data contains features with different scales, apply scaling techniques like Min-Max scaling or standardization.

Data Splitting:

Split our data into training, validation, and test sets. The training set is used for model training, the validation set for model tuning, and the test set for final model evaluation.

Data Preprocessing:

Apply any further data preprocessing steps such as standardization, dimensionality reduction, or other techniques that can improve the model's performance.

Data Export:

Save the cleaned and pre-processed data in a format that is easy to work with for building machine learning models. Common formats include CSV, Excel, or database tables. The quality and preparation of our data play a significant role in the success of our predictive analytics project. By completing this step effectively, we ensure that your data is ready for model development and that the models can learn meaningful patterns and make accurate predictions.

Step 3: Model Building:

1) Select Appropriate Algorithms: Choose machine learning algorithms that are suitable for our specific predictive analytics problem. Common choices include linear regression,

decision trees, random forests, support vector machines, or neural networks.

2)Data Preparation: Ensure our data is prepared and cleaned, as discussed in the earlier steps. This involves handling missing values, encoding categorical data, and scaling or normalizing features.

3)Split Data: Divide our dataset into three sets: a training set, a validation set, and a test set. The training set is used to train the model, the validation set is for tuning hyperparameters, and the test set is for final model evaluation.

4)Model Training: Train our chosen machine learning model on the training data.

Watson Studio provides an integrated environment to do this using popular libraries like scikit-learn or TensorFlow.

Step 4: Model Deployment :

Model deployment is a critical step in our project for becoming proficient in predictive analytics with IBM Cloud Watson Studio.

This step involves making your trained machine learning models accessible to applications through web services. Here's a guide on how to deploy your models:

1. Select the Model: Choose the best-performing machine learning model that you want to deploy. This should be the model that we have thoroughly trained and evaluated.

2. Preprocessing and Transformation: Ensure that any preprocessing steps (e.g., feature scaling or one-hot encoding) used during model training are replicated during deployment. This ensures that incoming data to the deployed model is processed correctly.

3. Create a Deployment Space:

- In IBM Watson Studio, navigate to the model deployment section.
- Create a deployment space or project where you can manage your deployment assets.

4.Model Deployment:

- In the deployment space, select the model we want to deploy.
- Follow the prompts to deploy the model. We may be asked to specify the deployment environment and configuration settings. Watson Studio makes this process user-friendly.

5. API Endpoint: Once deployed, our model will have a unique API endpoint that can be used to make predictions. This endpoint is typically a URL that accepts input data and

returns predictions.

6. Scalability and Load Balancing: Depending on our use case, consider setting up load balancing and ensuring the deployed model can handle a scalable number of requests.

7. Security and Access Control: Implement appropriate security measures to protect our API endpoint. Ensure that only authorized applications or users can access and use the model.

8. Documentation: Create documentation for the API endpoint. Include details about the expected input format, response format, and any required authentication or API keys.

9. Testing: Before integrating the model into applications, test the API endpoint to ensure it's functioning as expected. Use sample data to validate that it returns accurate predictions. Model deployment in IBM Cloud Watson Studio is designed

to be user-friendly and efficient. By following these steps, you ensure

that your trained models are accessible and capable of making realtime predictions, which is a fundamental part of unlocking datadriven insights for informed decision-making.

Step 5: Integration with Applications:

In this step, we'll seamlessly incorporate the deployed machine learning models into your existing or new software applications, such as web apps, mobile apps, or backend services. Integration enables these applications to utilize the predictive power of your models in real-time, which, in turn, empowers data-driven decision-making.

1. API Integration: Our deployed models typically expose an API endpoint that applications can communicate with. These APIs allow applications to send input data to the model and receive predictions as responses. Integration usually involves making HTTP requests to this API.

2. Data Input and Output Formats: Ensure that our applications understand the expected input data format for the model. This might involve converting user inputs into a format suitable for the model, and then processing the model's predictions.

3. Security: Implement security measures, including authentication and authorization, to ensure that only authorized applications can access and utilize your model's API. This helps protect sensitive data and the integrity of our predictive analytics.

4. Scalability: Consider the scalability of your application and model integration. Ensure that our applications can handle a high volume of requests and that our deployment

infrastructure can scale accordingly.

5. Error Handling: Implement robust error-handling mechanisms in our applications to deal with scenarios where the model may fail to make predictions or encounters unexpected input data.

6. Testing: Rigorously test the integration of the model within our applications using a variety of input data and scenarios. Verify that the predictions align with your expectations and are used appropriately.

By successfully integrating your predictive analytics models with our applications, we enable them to provide data-driven insights and real-time decision support to users. This is a pivotal step in making informed decisions, as outlined in your problem statement, and harnessing the magic of data-driven insights in a practical and actionable way.

Step 6: Monitoring and Maintenance:

To become proficient in predictive analytics with IBM Cloud Watson Studio, monitoring and maintenance are essential for ensuring that our predictive models consistently deliver accurate and valuable insights.

- **Metric Tracking:** Use specific performance metrics relevant to our problem. For instance, if we're predicting customer churn, track metrics like accuracy, precision, recall, or F1-score to evaluate the model's performance.
- **Continuous Performance Evaluation:** Regularly assess the performance of your deployed machine learning models. This involves measuring how well they make predictions and checking if they're still accurate.
- **Alert Systems:** Set up automated alerts or notifications that signal potential issues with your models. These alerts act as early warning systems, similar to smoke detectors in a house.
- **Data Updates:** If the data your models were trained on becomes outdated, consider updating it. New data ensures that your models remain relevant and effective, just like updating a map with new roads.
- **Periodic Retraining:** Depending on how rapidly your data evolves, periodic retraining of your models might be necessary. This process involves using fresh data to fine-tune the models for improved accuracy, akin to studying to improve your knowledge.
- **Documentation Updates:** Keep your documentation current. This documentation acts as a guide for anyone who manages or interacts with your models.

By the above steps, we can ensure that our predictive analytics capabilities remain up-to-date, accurate, and secure. Regular monitoring and maintenance are essential to unlock the full potential of data-driven insights and informed decision-making with IBM Cloud Watson Studio.

Real-Time Insights:

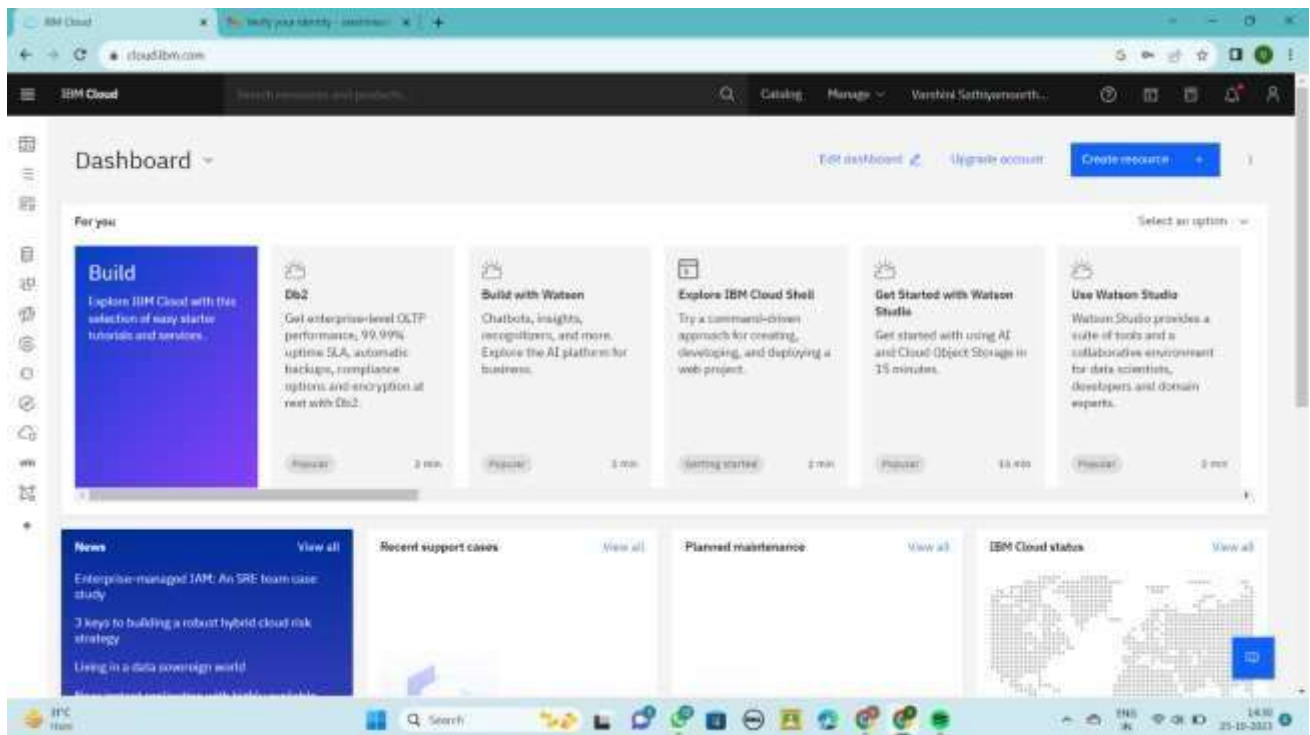
By deploying and integrating predictive models, we can gain access to real-time insights based on data. These insights help us to understand what's happening right now.

About IBM cloud Watson Studio:

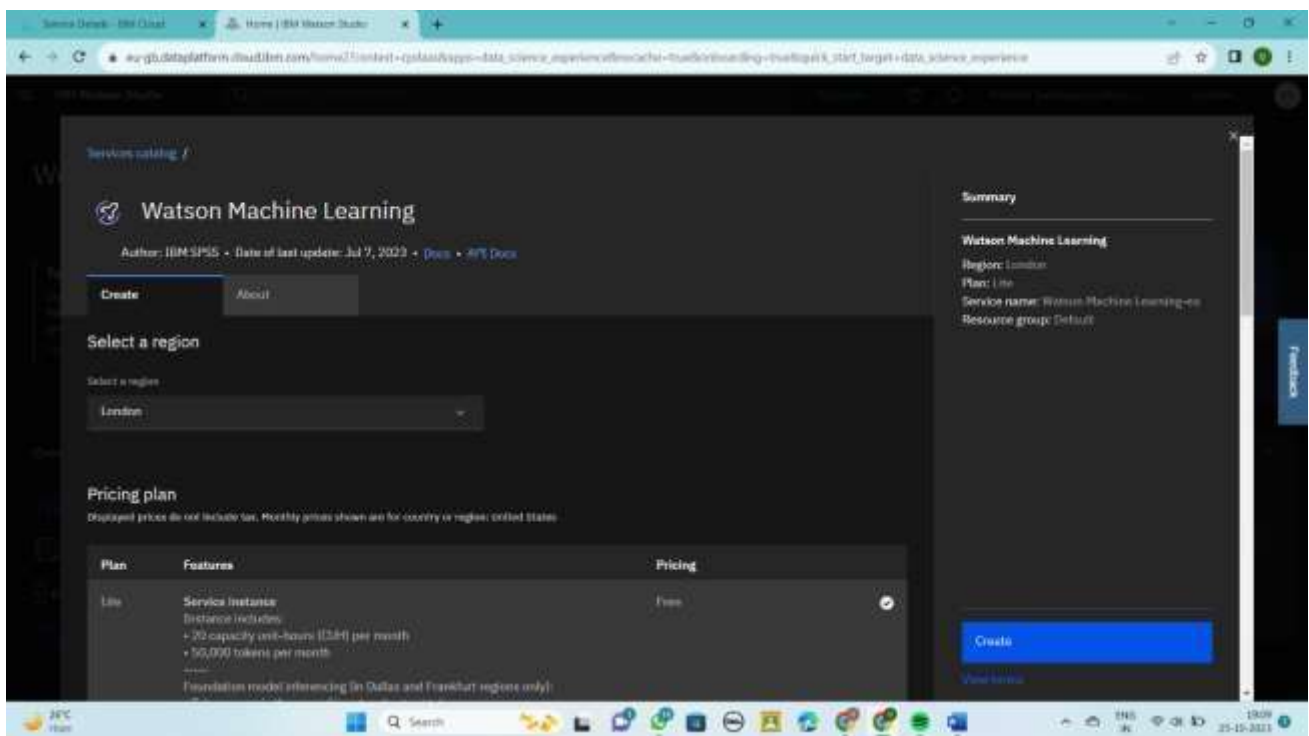
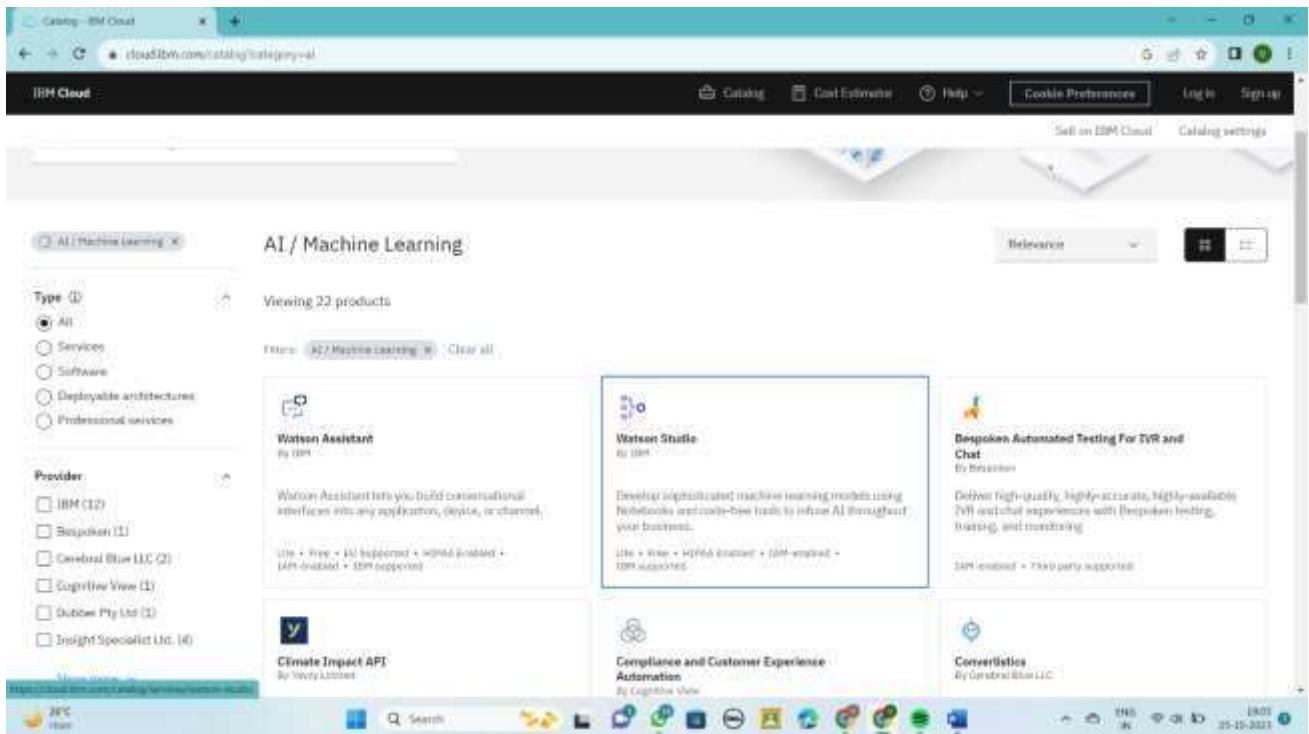
IBM Watson Studio provides tools for data scientists, application developers and subject matter experts to collaboratively and easily work with data to build and train models at scale. It gives us the flexibility to build models where our data resides and deploy anywhere in a hybrid environment so we can operationalize data science faster. IBM Watson Studio provides various tools for designing, training, and managing machine learning models: Model builder, Flow editor, Experiment builder, Notebooks, Machine learning command line interface. Among them, we use notebooks for the working with our dataset.

Stepwise procedure for building the machine learning model using IBM Cloud Watson Studio is as follows:

Step1: Login to IBM cloud



Step 2: Go to catalog and create a Watson Studio service in AI/Machine learning category.



Step 3: Create a project in IBM Watson Studio Dashboard and assign a Cloud object Storage service to manage datasets

Service Details - IBM Cloud

New project | IBM Watson Studio

+

ibm-gu.dataplatform.cloud.ibm.com/projects/create-project?context=cpd&id

IBM Watson Studio

Search by your workspaces

Upgrade

Varshini Saffiyamoorthy's ...

London

VS

New project

Define details

Name

Description (optional)

What's the purpose of this project?

Controls

☒ Restrict who can be a collaborator ⓘ

☐ Mark as sensitive ⓘ

Define storage

Project includes integration with [Cloud Object Storage](#) for storing project assets.

- 1 Select storage service

Add

Add an object storage instance, and then return to this page and click "Refresh".

Refresh

- 2 Refresh

Cancel

Create

Service Details - IBM Cloud

New project | IBM Watson Studio

Cloud Object Storage - Service

+

ibm-gu.dataplatform.cloud.ibm.com/data/catalog/cloud-object-storage?context=cpd&id=cloud-object-storage&id=cloud-object-storage&id=cloud-object-storage

IBM Watson Studio

Search by your workspaces

Upgrade

Varshini Saffiyamoorthy's ...

London

VS

Cloud Object Storage

Author: IBM • Date of last update: Jul 5, 2023 • Docs • API Docs

Create

About

Like plan services are deleted after 30 days of inactivity.

Standard

Standard plan is our most popular Pay-as-You-Go pricing plan. There is no minimum fee. This plan meets the requirements of most of the enterprise workloads.

[See pricing details](#)

Configure your resource

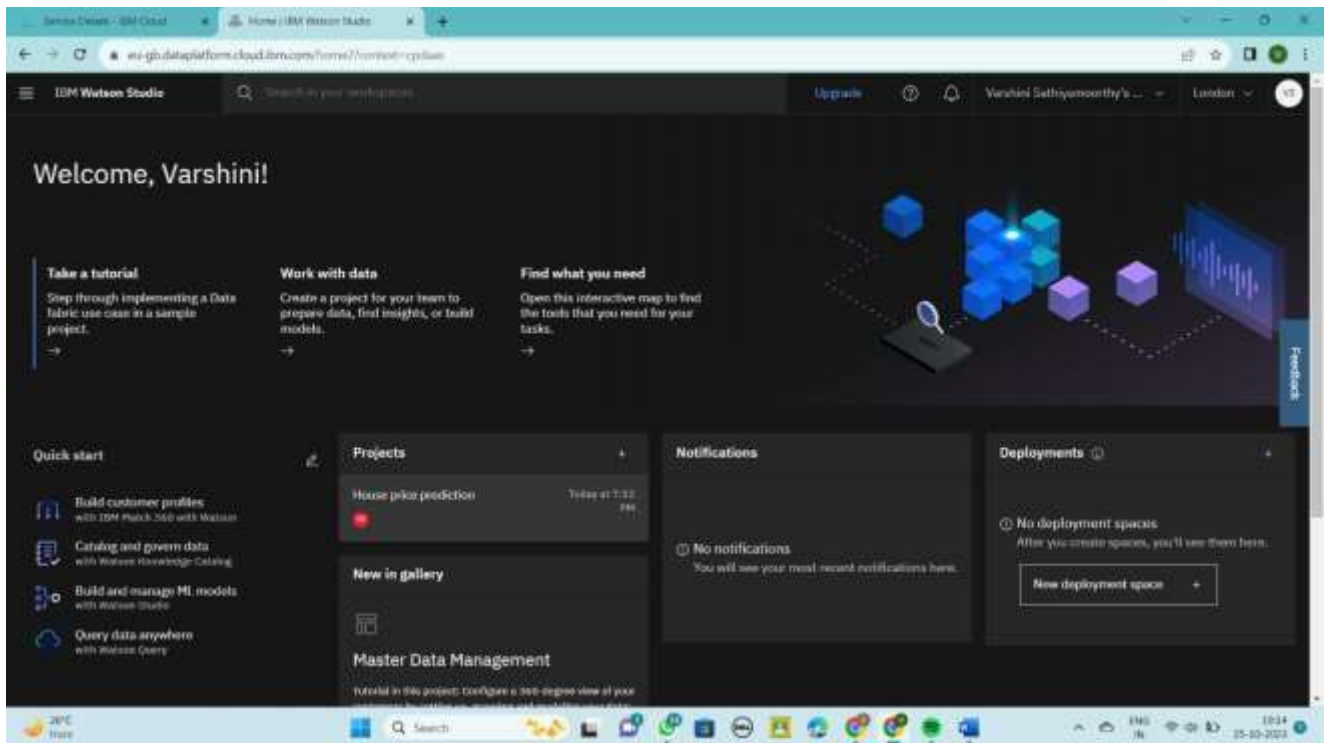
Service name

Select a resource group

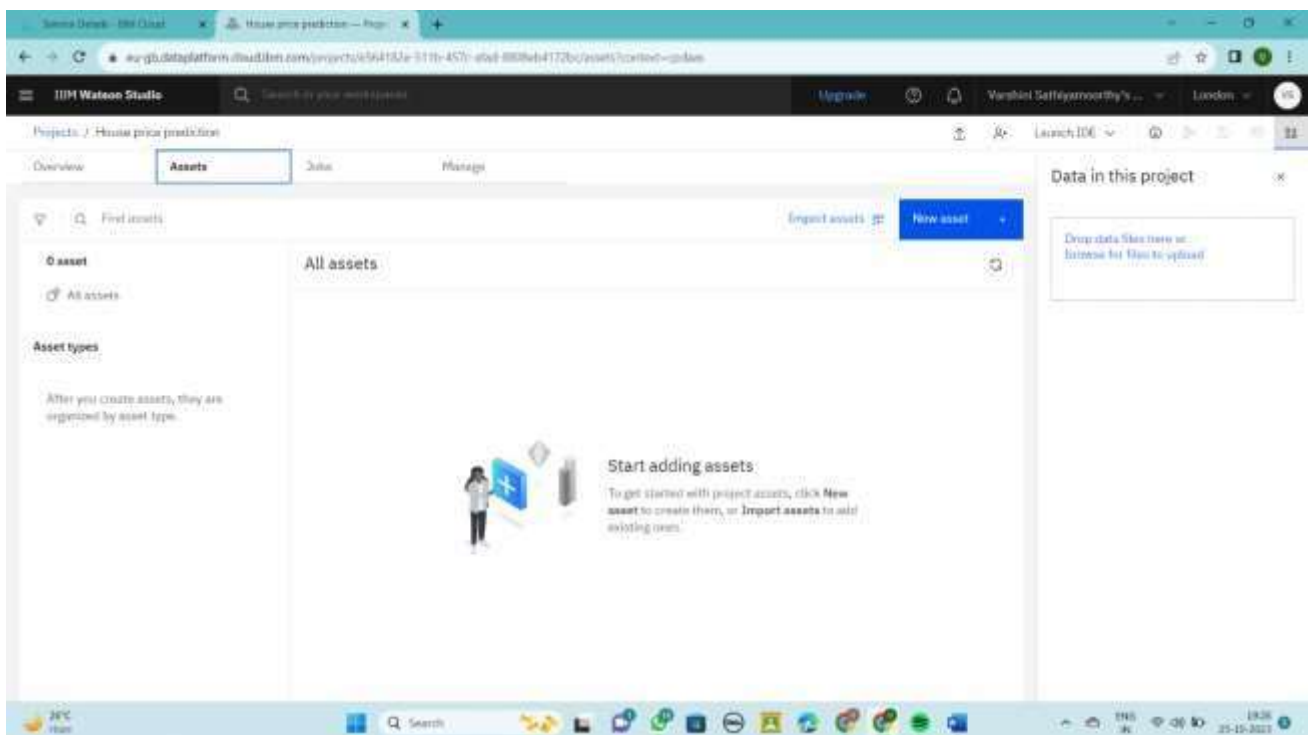
Tags

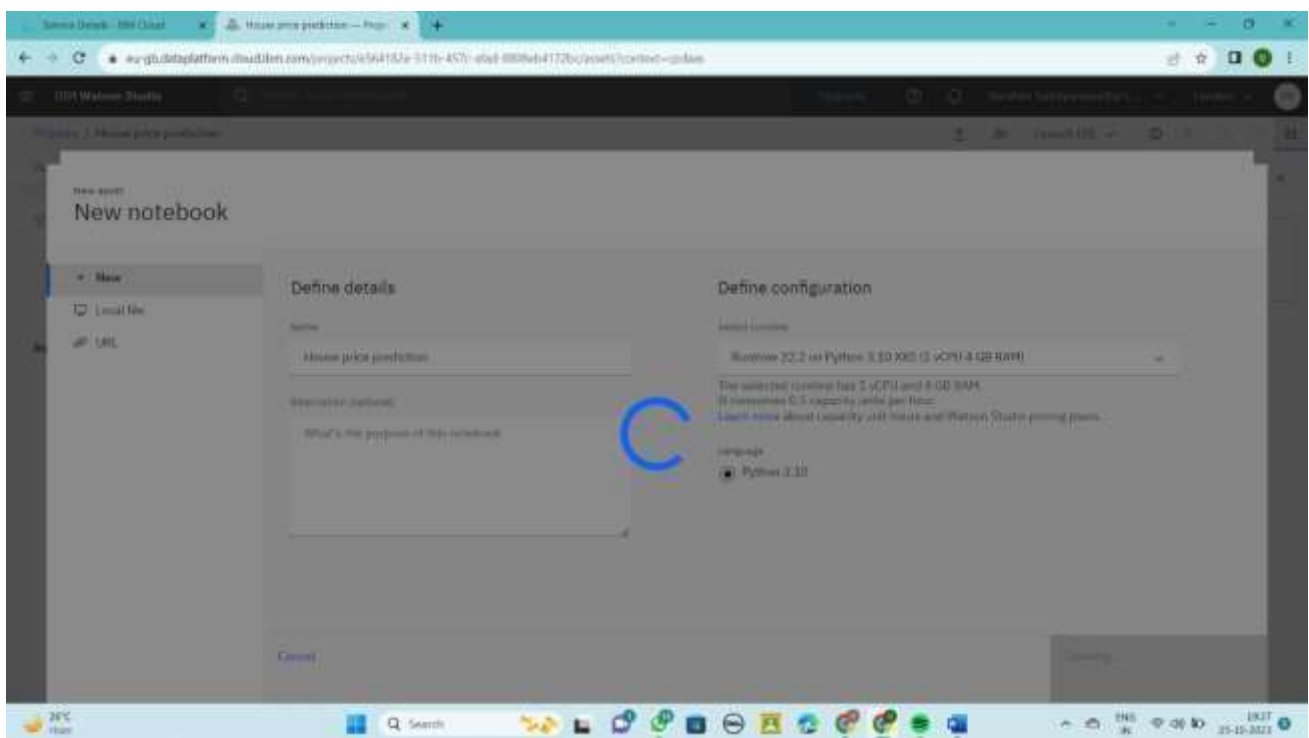
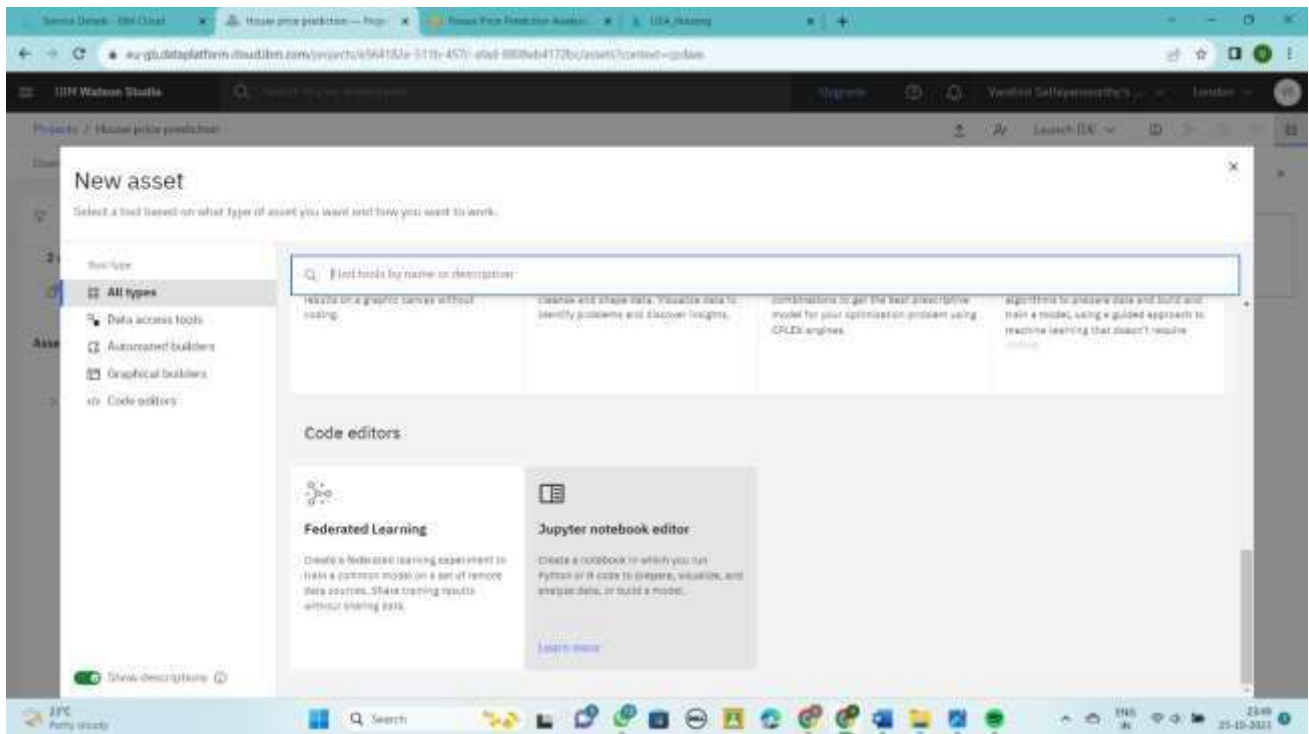
Creating...

[View terms](#)



Step 4: Add a jupyter notebook instance in your project to Develop and Deploy Machine Learning Model.





Step 5: Build a machine learning model using jupyter notebook instance.

Import the required libraries. In the next step we are going to upload and insert the dataset for training our Machine Learning model as pandas dataframe.

The screenshot shows the IBM Watson Studio interface. The notebook is titled 'House price prediction'. The code in the cell is as follows:

```
In [2]: import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split

In [3]: import os, types
import pandas as pd
from botocore.client import Config
import boto3

def __iter__(self): return 0

# hidden cell
# The following code accesses a file in your IBM Cloud Object Storage. It includes your credentials.
# You might want to remove those credentials before you share the notebook.
cos_client = boto3.client(service_name='s3',
                          aws_api_key_id='1',
                          aws_auth_endpoint='https://iam.cloud.ibm.com/oidc/token',
                          config=Config(signature_version='aws4'),
                          endpoint_url='https://s3.private.eu-gb.cloud-object-storage.appdomain.cloud')

bucket = 'housepriceprediction-ds0cfde4ta-pr-galltagpubw'
object_key = 'USA_housing.csv'

body = cos_client.get_object(Bucket=bucket, Key=object_key)['Body']
# add missing __iter__ method, so pandas accepts body as file-like object
if not hasattr(body, '__iter__'): body.__iter__ = types.MethodType(__iter__, body)

df = pd.read_csv(body)
df.head()
```

On the right side, the 'Read data' panel shows the detected data as 'USA_housing.csv' and the load as 'pandas DataFrame'.

The screenshot shows the same IBM Watson Studio notebook. The code in the cell is as follows:

```
bucket = 'housepriceprediction-ds0cfde4ta-pr-galltagpubw'
object_key = 'USA_housing.csv'

body = cos_client.get_object(bucket=bucket, Key=object_key)['Body']
# add missing __iter__ method, so pandas accepts body as file-like object
if not hasattr(body, '__iter__'): body.__iter__ = types.MethodType(__iter__, body)

dataset = pd.read_csv(body)
dataset.head()
```

The output of the code is displayed below the cell:

```
Out[4]:
```

	Avg. Area Income	Avg. Area House Age	Avg. Area Number of Rooms	Avg. Area Number of Bedrooms	Area Population	Price	Address
0	70643.45024	5.582661	7.909188	4.99	13000.800603	1.990034e+06	200 Manuel Ferry Apt. 6700, Westbury, NE 3701
1	75048.94048	6.022600	6.730201	3.99	42173.072174	1.302881e+06	188 Johnson Views Suite 0700, San Rafael, CA
2	81287.367179	5.885680	8.512727	5.33	38832.159431	1.355686e+06	9127 Elizabeth Stovers Rd, Danvers, WI 09482
3	83345.240046	7.183236	5.586729	3.25	24010.242831	1.250017e+06	1555 Barak Rd P.O. Box 44020
4	69932.917228	5.540558	7.839588	4.23	28264.106472	8.300435e+05	1890 Raymond P.O. Box 99388

Once the dataset is imported we can proceed further with pre-processing steps and building the model as follows.

The screenshot shows a Jupyter Notebook in IBM Watson Studio. The top part of the notebook displays a table with 5 rows of house data. The bottom part shows the output of a `dataset.isnull().any()` command, which returns a list of features and their null status.

	7564545824	5.582961	7.99188	4.29	2266.30533	1.55634e+05	339 Michael Perry Apt. 8766Lacrosse, NC 2761
1	75248942455	6.502900	6.73001	3.39	45172.072174 <th>1.50568e+05</th> <th>188 Johnson Vines Suite 0797Lakeside, CA</th>	1.50568e+05	188 Johnson Vines Suite 0797Lakeside, CA
2	81287987178	5.388880	8.512727	5.13	36932.159430	1.05088e+05	9127 Elizabeth StevensLnDarien, WI 53003
3	83345340048	7.180238	5.98729	3.29	34310.242831	1.25061e+05	US5-BarnettFPO AP 4402
4	69882397228	5.540555	7.335888	4.23	28354.106472	6.309435e+05	US98 RaymondFPO AE 9888

```
In [5]: dataset.isnull().any()

Out[5]: Avg. Area Income      False
Avg. Area House Age      False
Avg. Area Number of Rooms  False
Avg. Area Number of Bedrooms False
Area Population          False
Price                   False
Address                 False
dtype: bool
```

Splitting into training and testing datasets

The screenshot shows a Jupyter Notebook in IBM Watson Studio. The notebook contains code to split the dataset into training and testing sets using `train_test_split` from `sklearn.model_selection`.

```
In [5]: dataset.isnull().any()

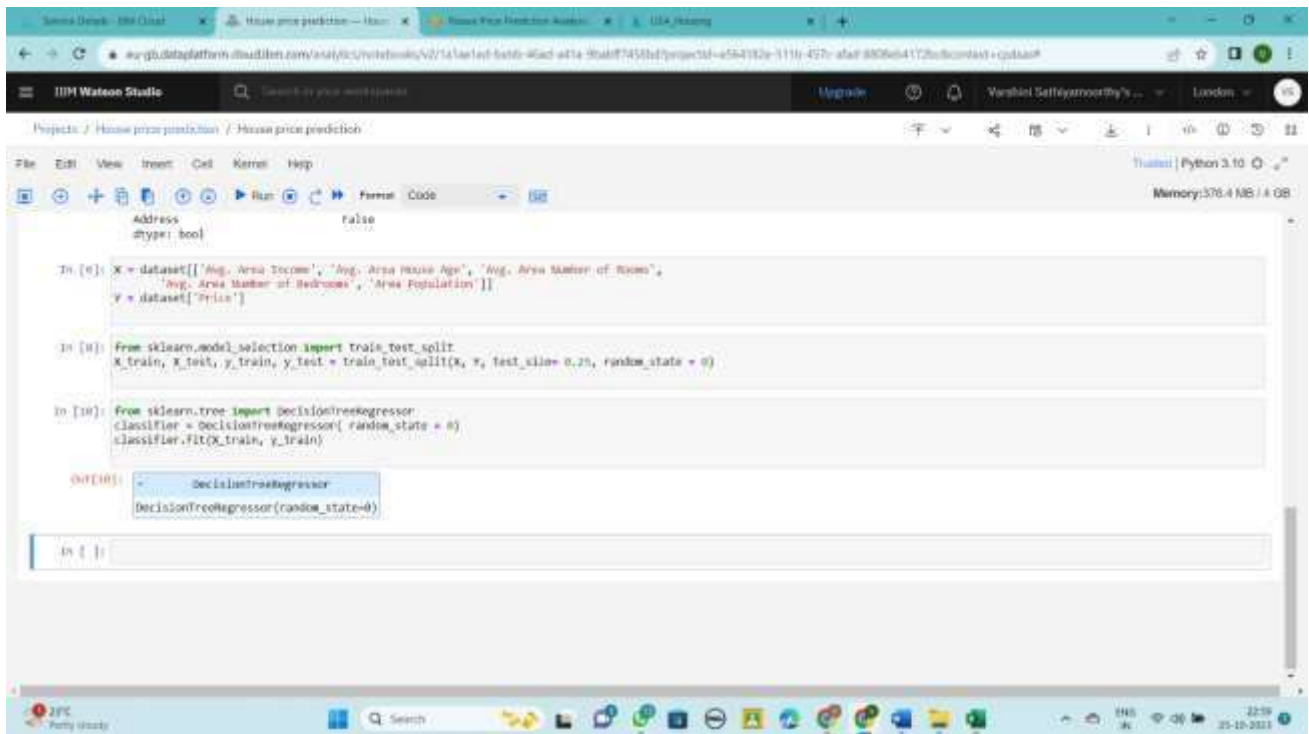
Out[5]: Avg. Area Income      False
Avg. Area House Age      False
Avg. Area Number of Rooms  False
Avg. Area Number of Bedrooms False
Area Population          False
Price                   False
Address                 False
dtype: bool

In [6]: X = dataset[['Avg. Area Income', 'Avg. Area House Age', 'Avg. Area Number of Rooms',
                    'Avg. Area Number of Bedrooms', 'Area Population']]
        y = dataset['Price']

In [8]: from sklearn.model_selection import train_test_split
        x_train, x_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=0)

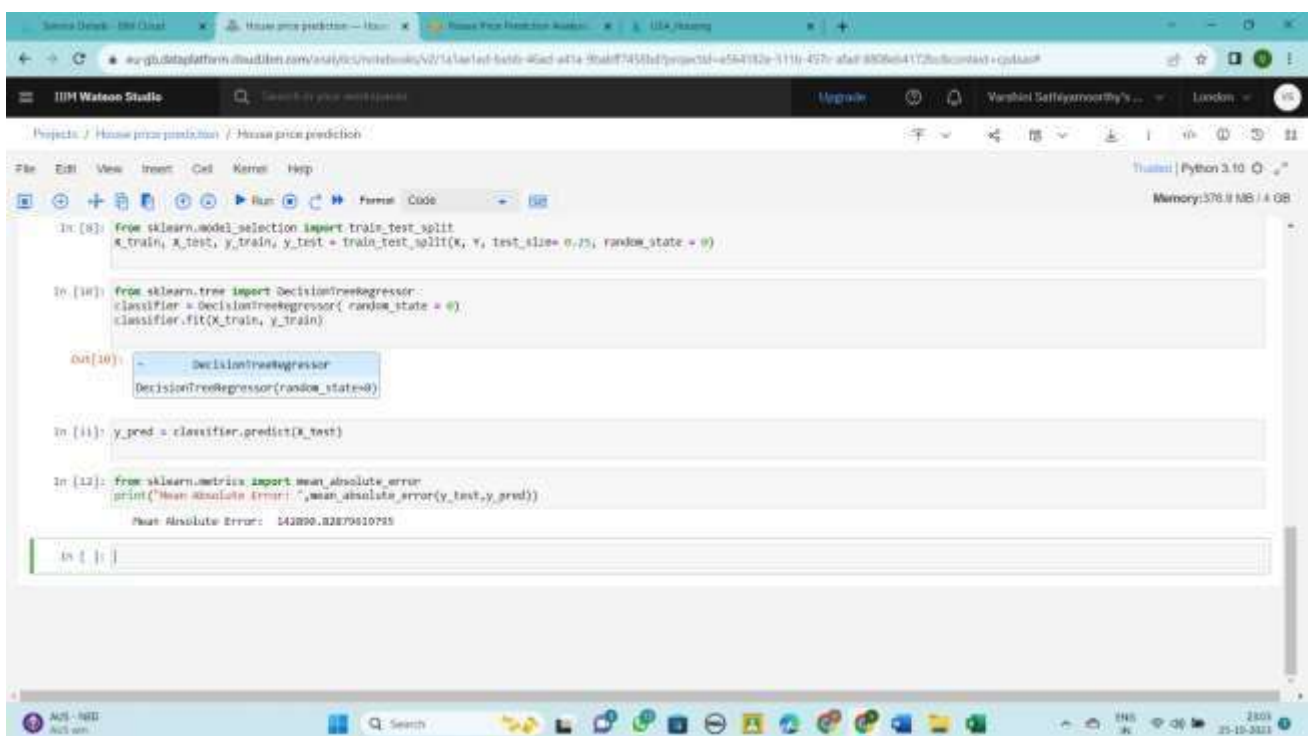
In [ ]:
```

Fitting decision tree regression to trained model:



```
In [6]: X = dataset[['Avg. Area Total', 'Avg. Area House Age', 'Avg. Area Number of Rooms',  
                'Avg. Area Number of Bedrooms', 'Area Population']]  
y = dataset['Price']  
  
In [8]: from sklearn.model_selection import train_test_split  
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=0)  
  
In [10]: from sklearn.tree import DecisionTreeRegressor  
classifier = DecisionTreeRegressor(random_state=0)  
classifier.fit(X_train, y_train)  
  
Out[10]: DecisionTreeRegressor  
DecisionTreeRegressor(random_state=0)
```

Predicting the Test set results and finding the mean absolute error



```
In [8]: from sklearn.model_selection import train_test_split  
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=0)  
  
In [10]: from sklearn.tree import DecisionTreeRegressor  
classifier = DecisionTreeRegressor(random_state=0)  
classifier.fit(X_train, y_train)  
  
Out[10]: DecisionTreeRegressor  
DecisionTreeRegressor(random_state=0)  
  
In [11]: y_pred = classifier.predict(X_test)  
  
In [12]: from sklearn.metrics import mean_absolute_error  
print("Mean Absolute Error: ", mean_absolute_error(y_test, y_pred))  
  
Mean Absolute Error: 141890.82879810795
```

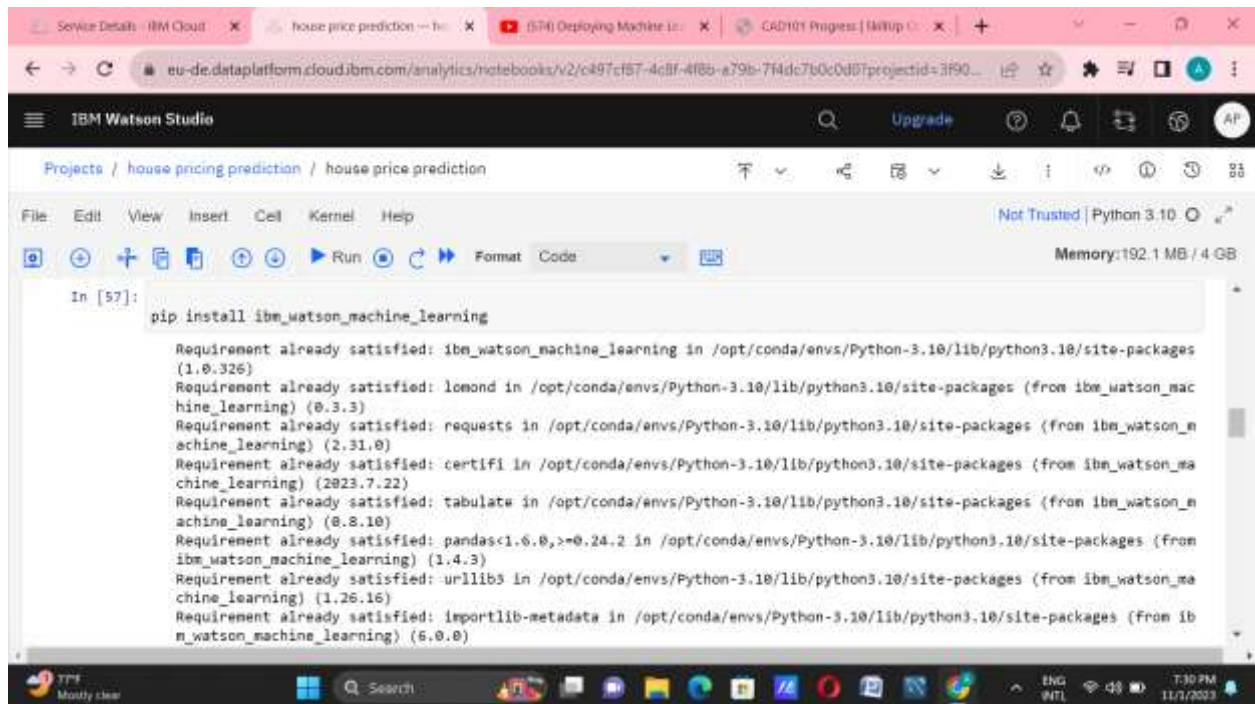
Thus the machine learning model was built, trained and tested using IBM Watson Studio.

Deployment of machine learning model and creation of web application for our machine learning model

Step:1

Install `ibm_watson_machine_learning` by the command

```
#pip install ibm_watson_machine_learning
```

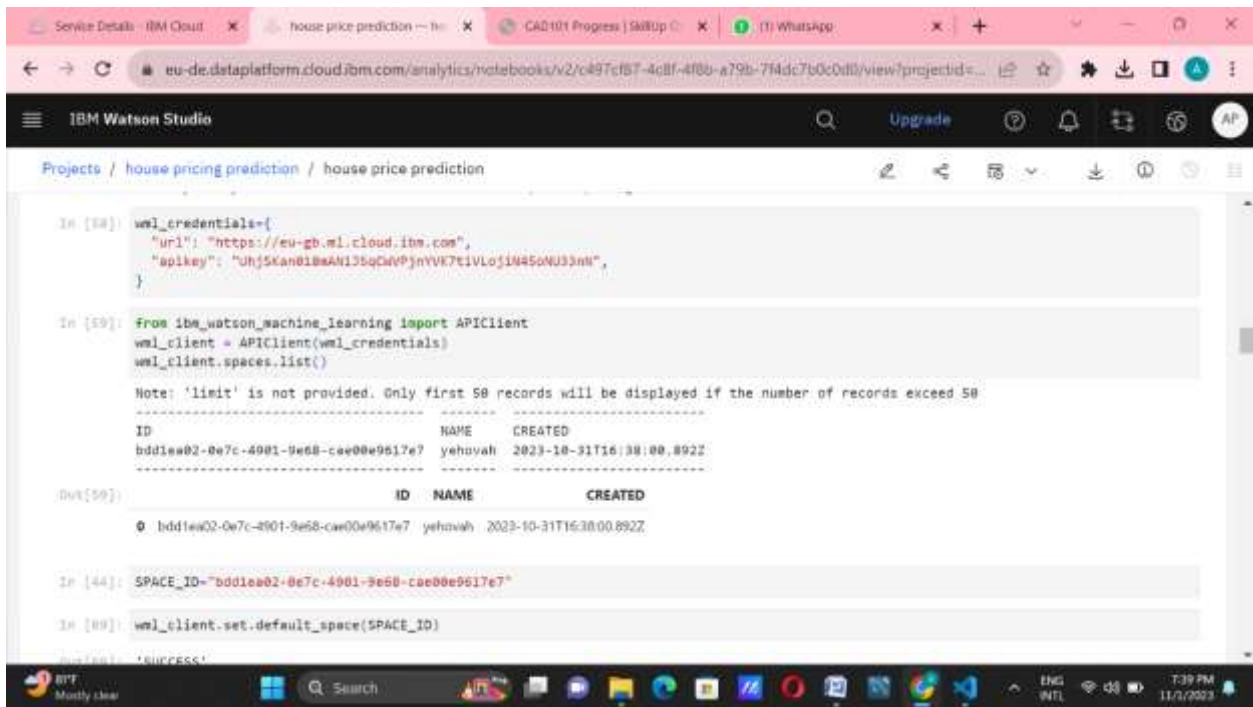


The screenshot shows the IBM Watson Studio web interface. The browser address bar displays a URL from `eu-de.dataplatform.cloud.ibm.com`. The interface includes a top navigation bar with 'IBM Watson Studio' and an 'Upgrade' button. Below this is a breadcrumb trail: 'Projects / house pricing prediction / house price prediction'. The main area is a code editor with a menu bar (File, Edit, View, Insert, Cell, Kernel, Help) and a toolbar with icons for file operations, running, and formatting. The code editor shows the command `pip install ibm_watson_machine_learning` in a cell. The output of the command is displayed below the code, showing that all required dependencies are already satisfied in the current environment. The dependencies listed include `ibm_watson_machine_learning` (1.0.326), `lcmomd` (0.3.3), `requests` (2.31.0), `certifi` (2023.7.22), `tabulate` (0.8.10), `pandas` (1.6.0), `urllib3` (1.26.16), and `importlib-metadata` (6.0.0). The bottom of the screen shows a Windows taskbar with various application icons and a system clock indicating 11:30 PM on 11/7/2023.

```
In [57]: pip install 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)
Requirement already satisfied: lcmomd in /opt/conda/envs/Python-3.10/lib/python3.10/site-packages (from ibm_watson_machine_learning) (0.3.3)
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: certifi in /opt/conda/envs/Python-3.10/lib/python3.10/site-packages (from ibm_watson_machine_learning) (2023.7.22)
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: 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: urllib3 in /opt/conda/envs/Python-3.10/lib/python3.10/site-packages (from ibm_watson_machine_learning) (1.26.16)
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)
```

Step 2: Import `WatsonMachineLearningAPIClient` library. Watson studio uses Watson Machine Learning service credentials to access WML service, so paste the credentials.



```
In [38]: wml_credentials={
        "url": "https://eu-gb.ml.cloud.ibm.com",
        "apikey": "Uhj5Kan818aAN135qDwVpJnYVK7t1VLoj1N456NU33nH",
    }

In [39]: from ibm_watson_machine_learning import APIClient
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
-----
bdd1ea02-0e7c-4001-9e68-cae00e9617e7 yehovah 2023-10-31T16:38:00.892Z

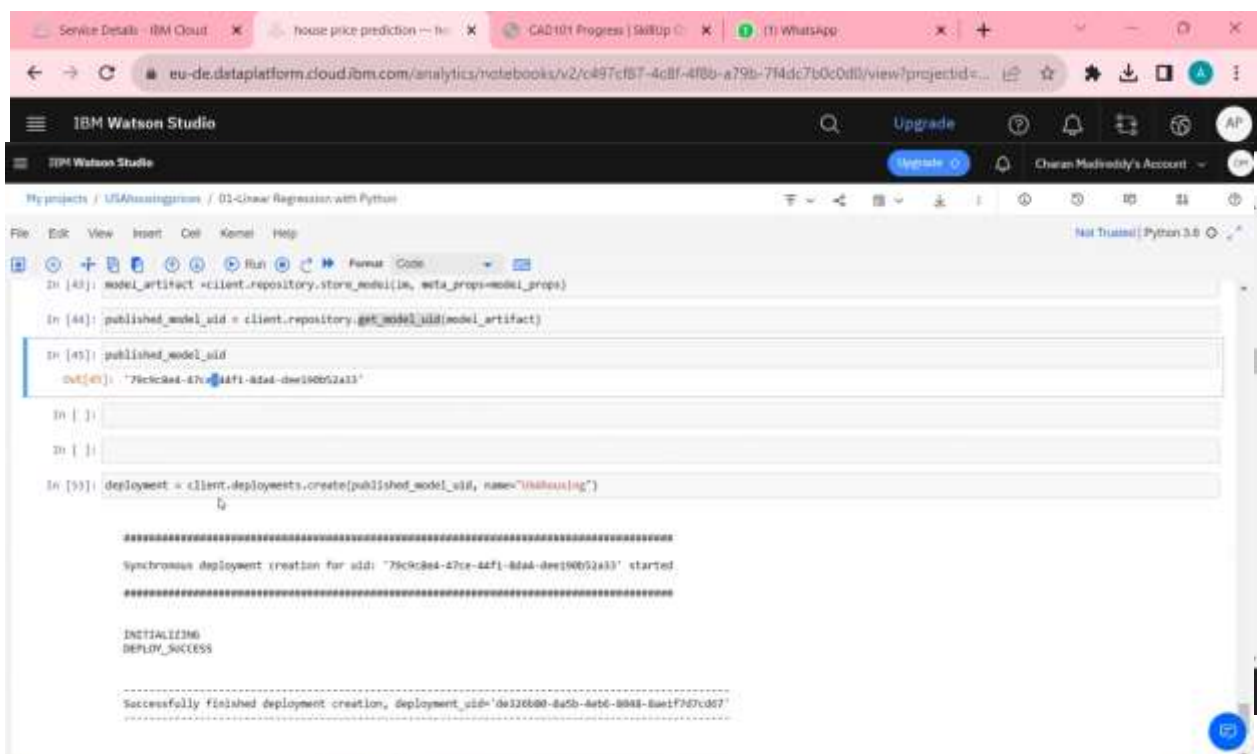
Out[39]: ID          NAME          CREATED
0 bdd1ea02-0e7c-4001-9e68-cae00e9617e7 yehovah 2023-10-31T16:38:00.892Z

In [44]: SPACE_ID="bdd1ea02-0e7c-4001-9e68-cae00e9617e7"

In [89]: wml_client.set.default_space(SPACE_ID)

Out[89]: 'SUCCESS!'
```

Step-3: In this step we have to specify our machine learning model properties and store the model in WML repository.



```
In [43]: model_artifact = client.repository.store_model(ma, wml_props=model_props)

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

In [45]: published_model_uid
Out[45]: "79c9c8d4-47ce-44f1-8d44-dde190b52a33"

In [ ]:

In [ ]:

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

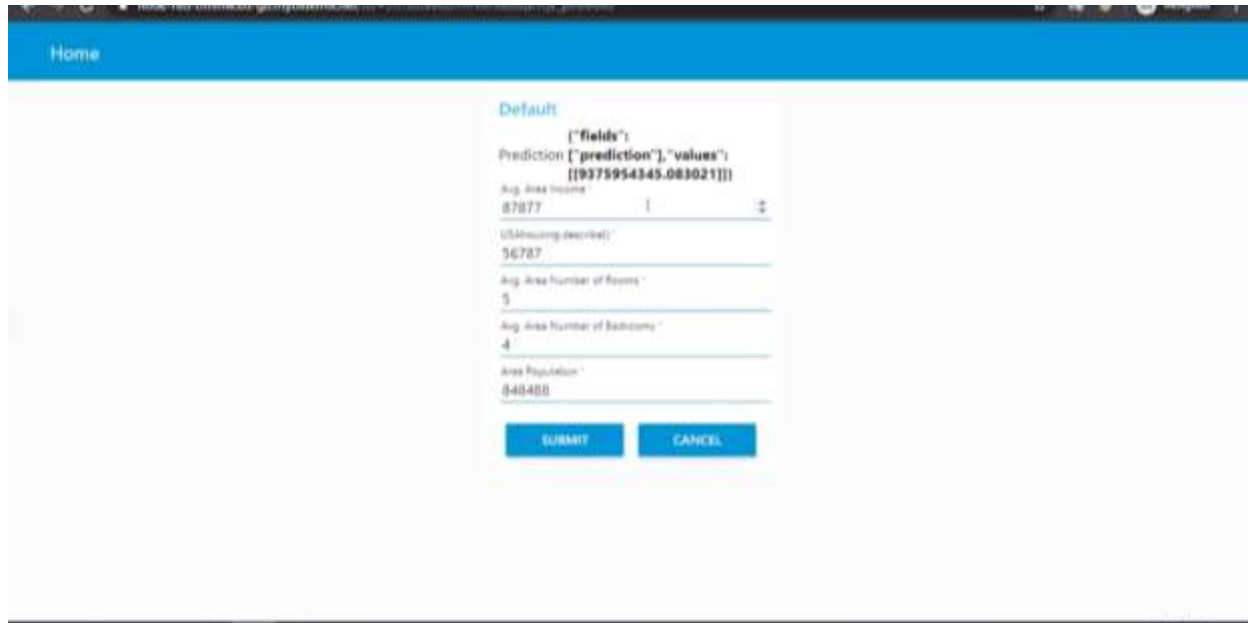
=====
Synchronous deployment creation for uid: "79c9c8d4-47ce-44f1-8d44-dde190b52a33" started
=====

INITIALIZING
DEPLOY_SUCCESS

=====
Successfully finished deployment creation, deployment_uid="da326b00-8a5b-4eb6-b948-8ae1f767c067"
=====
```

Step :4 Successfully we have deployed our model in IBM Watson Studio.

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 displays a web application interface with a blue header bar labeled "Home". Below the header, there is a "Default" section. This section contains a prediction result and several input fields for features. The prediction result is displayed as a JSON object: `{ "prediction": 19375954345.083021 }`. The input fields are labeled "Avg Area Income", "USHousehold Income", "Avg Area Number of Rooms", "Avg Area Number of Bedrooms", and "Area Population". Each field has a corresponding input value. At the bottom of the form, there are two buttons: "SUBMIT" and "CANCEL".

Field	Value
Prediction	19375954345.083021
Avg Area Income	87877
USHousehold Income	56787
Avg Area Number of Rooms	5
Avg Area Number of Bedrooms	4
Area Population	840488

Thus the deployed model can be accessed and utilized for real-time predictions.