

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. Unlock the magic of data-driven insights and make informed decisions like never before!

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

Design thinking is a human-centered approach to solving problems and creating innovative solutions. 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**

Datasetselection: (“ house-price-prediction”)

- 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:

The House Price Prediction Project has to be deployed using a decision tree.

Decision Trees are a foundational machine learning algorithm with many variations and extensions (e.g., Random Forests,

Gradient Boosting Trees) that have improved their performance and robustness. They are widely used in various domains due to their simplicity and interpretability.

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.

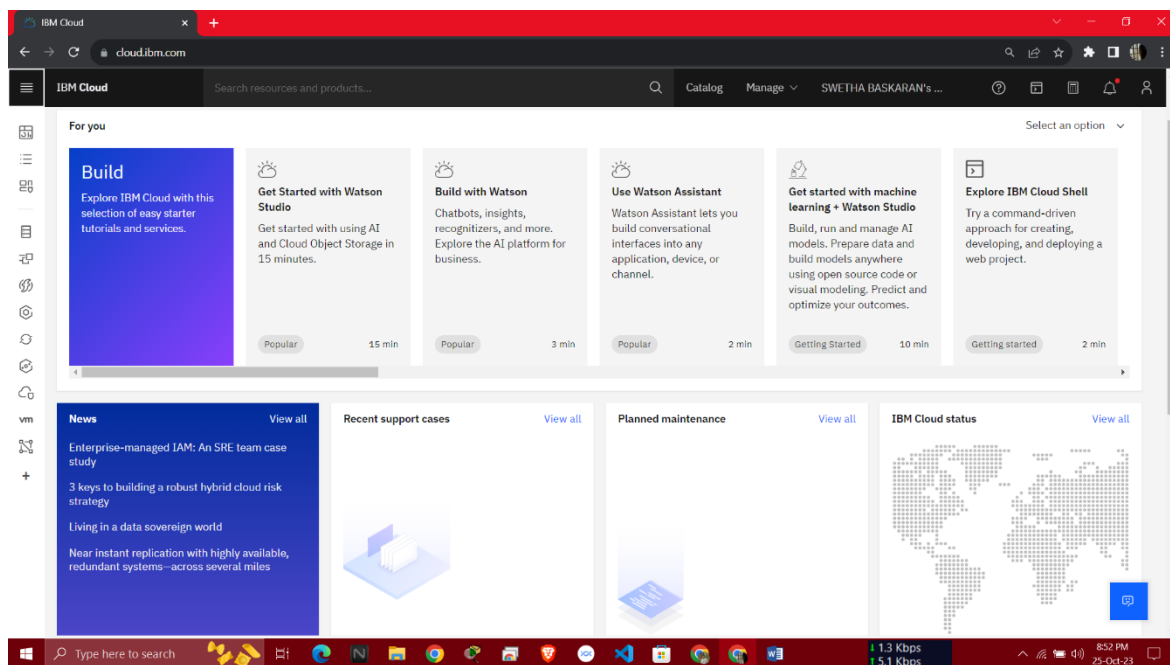
Real-Time Insights:

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.

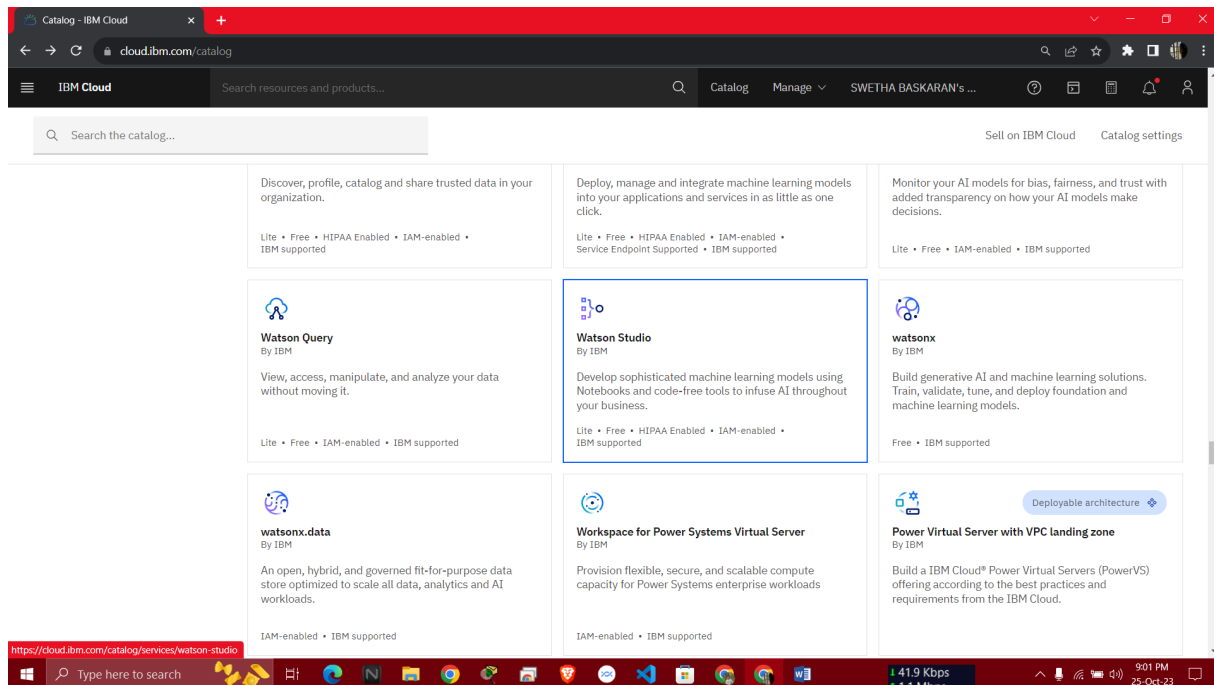
Modeling and Evaluation:

Step wise process for the House Price Prediction Analysis Machine learning model deployment.

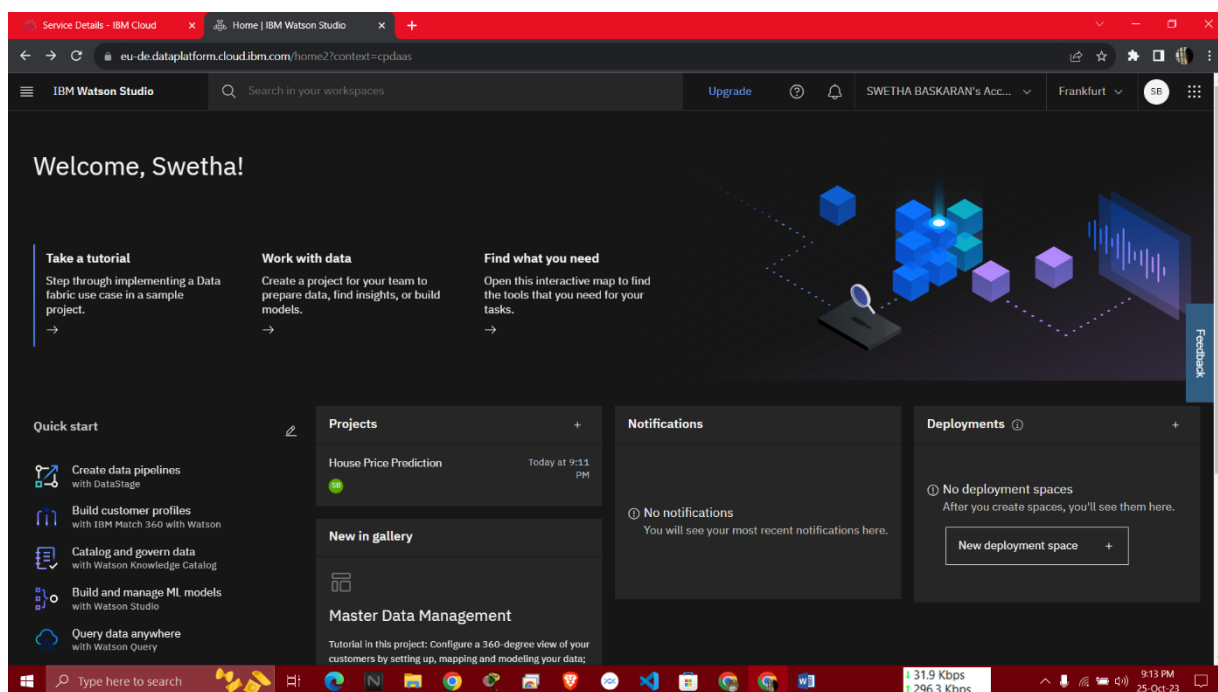
Step 1: Login to IBM cloud



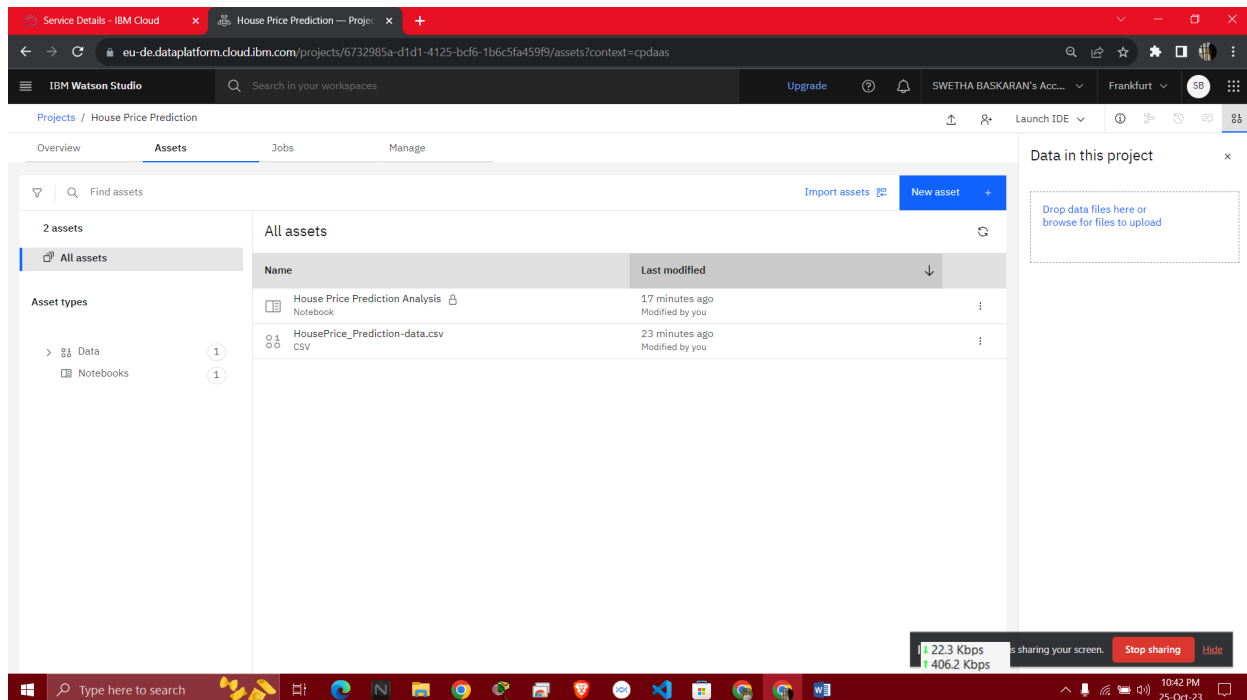
Step 2: Go to catalog and create a Watson Studio service in AI category.



Step 3: Get started to launch Watson Studio Dashboard.



Step 4: Create a project in IBM Watson Studio in IBM Cloud and assign a Cloud object storage to manage datasets.



Cloud Object Storage is a storage service in IBM Cloud. We use this service to manage our datasets for training the ML Model and store required files.

Step 5: Add a jupyter notebook instance in your project to Develop and Deploy Machine Learning Model.

- i. Import necessary library packages.
 - import pandas as pd: For handling your dataset.
 - import numpy as np: Useful for numerical operations.
 - from sklearn.model_selection import train_test_split: Split your data into training and testing sets.
 - import seaborn as sns: for data visualization in python.

- import matplotlib.pyplot as plt: another powerful library for creating visualizations.

```

In [1]: 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
from sklearn.preprocessing import StandardScaler

In [2]: import os, types
import pandas as pd
from boto3.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',
    ibm_api_key_id='C09rS190qj5j9Mo7IP4p58dwlFKP0zjnpVLURY1uKPH',
    ibm_auth_endpoint='https://iam.cloud.ibm.com/oidc/token',
    config=Config(signature_version='oauth'),
    endpoint_url='https://s3.private.eu-de.cloud-object-storage.appdomain.cloud')

bucket = 'housepricepredictionanalytics-donotdelete-pr-ltw9y3mxzbmh2p'
object_key = 'HousePrice_Prediction-data.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)

```

- ii. Import dataset and proceed further with pre-processing steps and build the model.

```

In [2]: import os, types
import pandas as pd
from boto3.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',
    ibm_api_key_id='C09rS190qj5j9Mo7IP4p58dwlFKP0zjnpVLURY1uKPH',
    ibm_auth_endpoint='https://iam.cloud.ibm.com/oidc/token',
    config=Config(signature_version='oauth'),
    endpoint_url='https://s3.private.eu-de.cloud-object-storage.appdomain.cloud')

bucket = 'housepricepredictionanalytics-donotdelete-pr-ltw9y3mxzbmh2p'
object_key = 'HousePrice_Prediction-data.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)

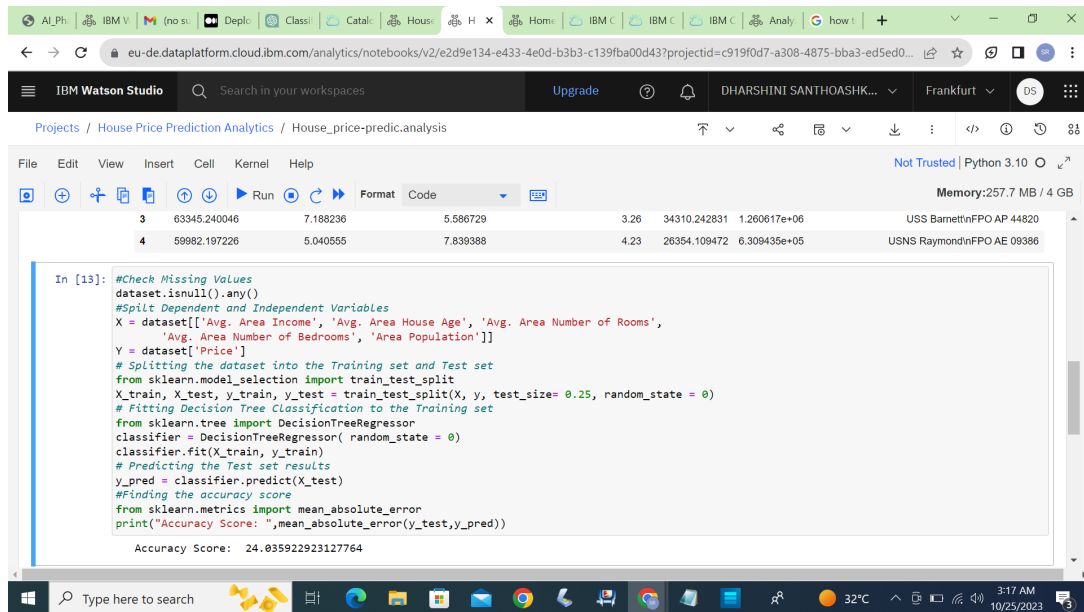
dataset = pd.read_csv(body)
dataset.head()

Out[2]:

```

	Avg. Area Income	Avg. Area House Age	Avg. Area Number of Rooms	Avg. Area Number of Bedrooms	Area Population	Price	Address
0	79545.426574	5.682861	7.009188	4.09	23066.800003	1.059034e+06	208 Michael Ferry Apt. 674inLaurabury, NE 3701...
1	79548.642455	6.002900	6.730821	3.09	40173.072174	1.505891e+06	188 Johnson Views Suite 079inLake Kathleen, CA...
2	61287.067179	5.865890	8.512727	5.13	36882.159400	1.058988e+06	9127 Elizabeth StaveneninDaneiloom, WI 06482...
3	63345.240046	7.188236	5.586729	3.26	34310.242831	1.260617e+06	USS BarnettinFPO AP 44820
4	59982.197226	5.040555	7.839388	4.23	26354.109472	6.309430e+05	USNS RaymondinFPO AE 09386

Step 6: Train the build model and evaluate them.



The screenshot shows the IBM Watson Studio interface. At the top, there's a browser window with the URL `eu-de.dataplatform.cloud.ibm.com/analytics/notebooks/v2/e2d9e134-e433-4e0d-b3b3-c139fba00d43?projectId=c919f0d7-a308-4875-bba3-ed5ed0...`. Below the browser, the IBM Watson Studio header is visible with the user name 'DHARSHINI SANTHOASHK...' and the location 'Frankfurt'. The main workspace shows a notebook titled 'House Price Prediction Analytics' with a sub-tab 'House_price-predic.analysis'. The notebook content includes a table of data and a code cell.

3	63345.240046	7.188236	5.586729	3.26	34310.242831	1.260617e+06	USS BarnettinFPO AP 44820
4	59982.197226	5.040555	7.839388	4.23	26354.109472	6.309435e+05	USNS RaymondinFPO AE 09386

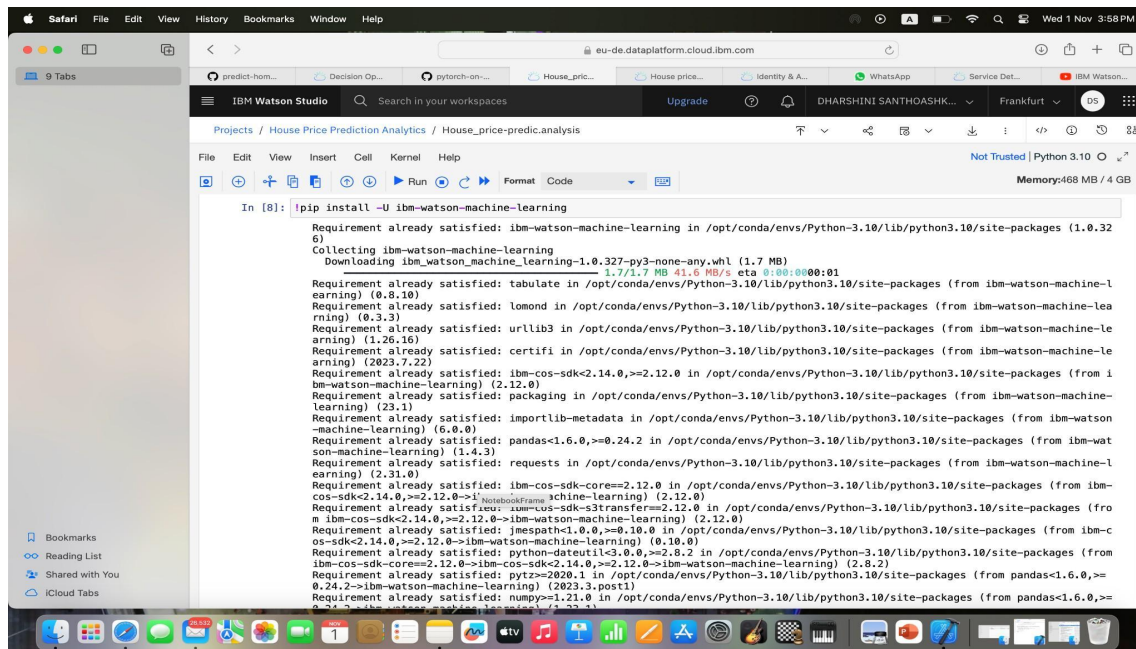
```
In [13]: #Check Missing Values
dataset.isnull().any()
#Split Dependent and Independent Variables
X = dataset[['Avg. Area Income', 'Avg. Area House Age', 'Avg. Area Number of Rooms',
            'Avg. Area Number of Bedrooms', 'Area Population']]
Y = dataset['Price']
# Splitting the dataset into the Training set and Test set
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)
# Fitting Decision Tree Classification to the Training set
from sklearn.tree import DecisionTreeRegressor
classifier = DecisionTreeRegressor( random_state = 0)
classifier.fit(X_train, y_train)
# Predicting the Test set results
y_pred = classifier.predict(X_test)
#Finding the accuracy score
from sklearn.metrics import mean_absolute_error
print("Accuracy Score: ",mean_absolute_error(y_test,y_pred))

Accuracy Score: 24.035922923127764
```

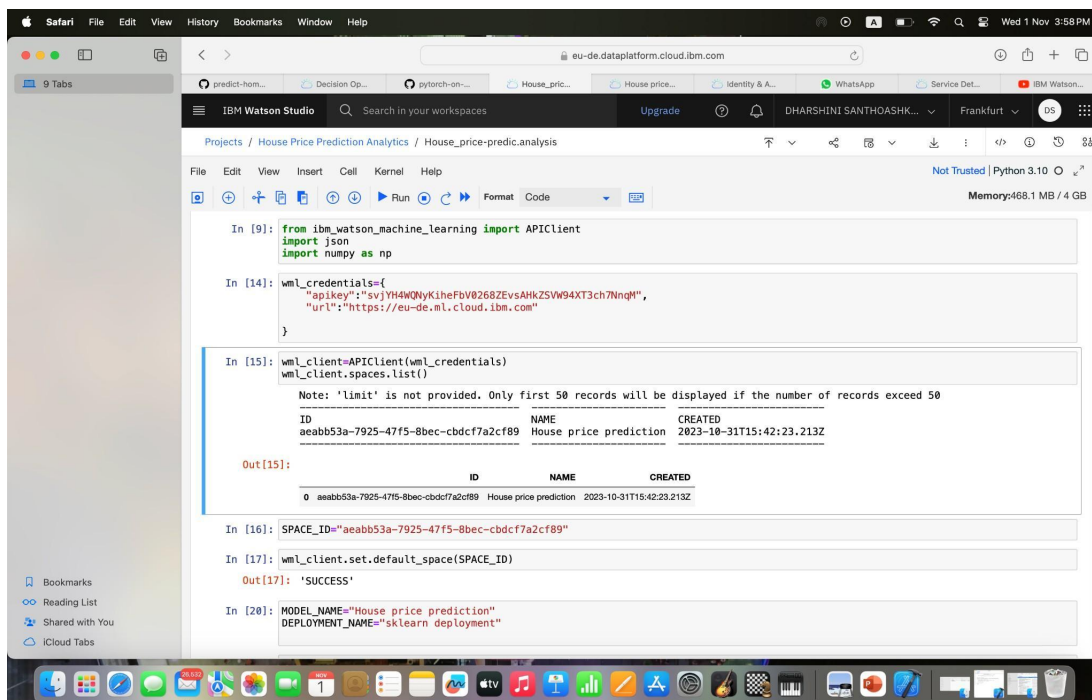
Model was build trained and tested

Project Deployment:

Step 1: Installing the required libraries



Step 2: Adding Watson Machine Learning service credentials to access



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.
 - 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)
- Managing model deployments (e.g., starting, stopping, deleting)

5. Implement data preprocessing:

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

6. Testing:

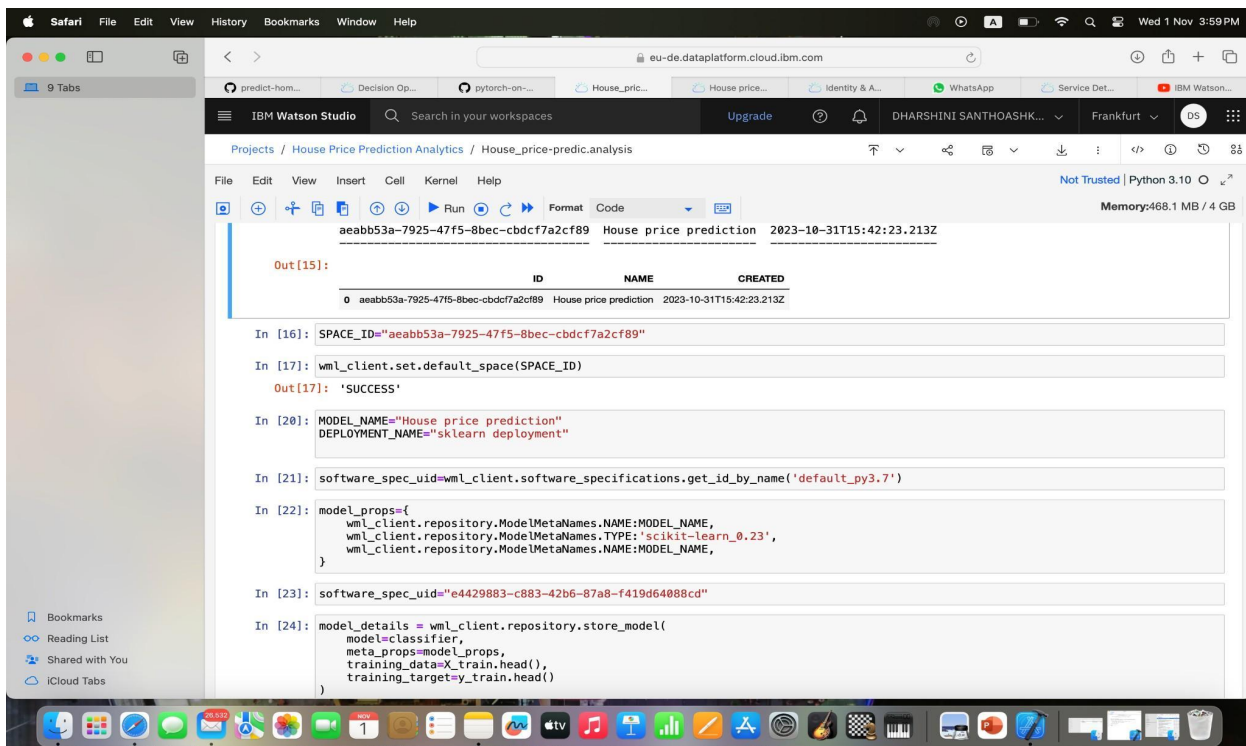
- Create unit tests for your client library to ensure that it functions as expected.

7. Usage examples:

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

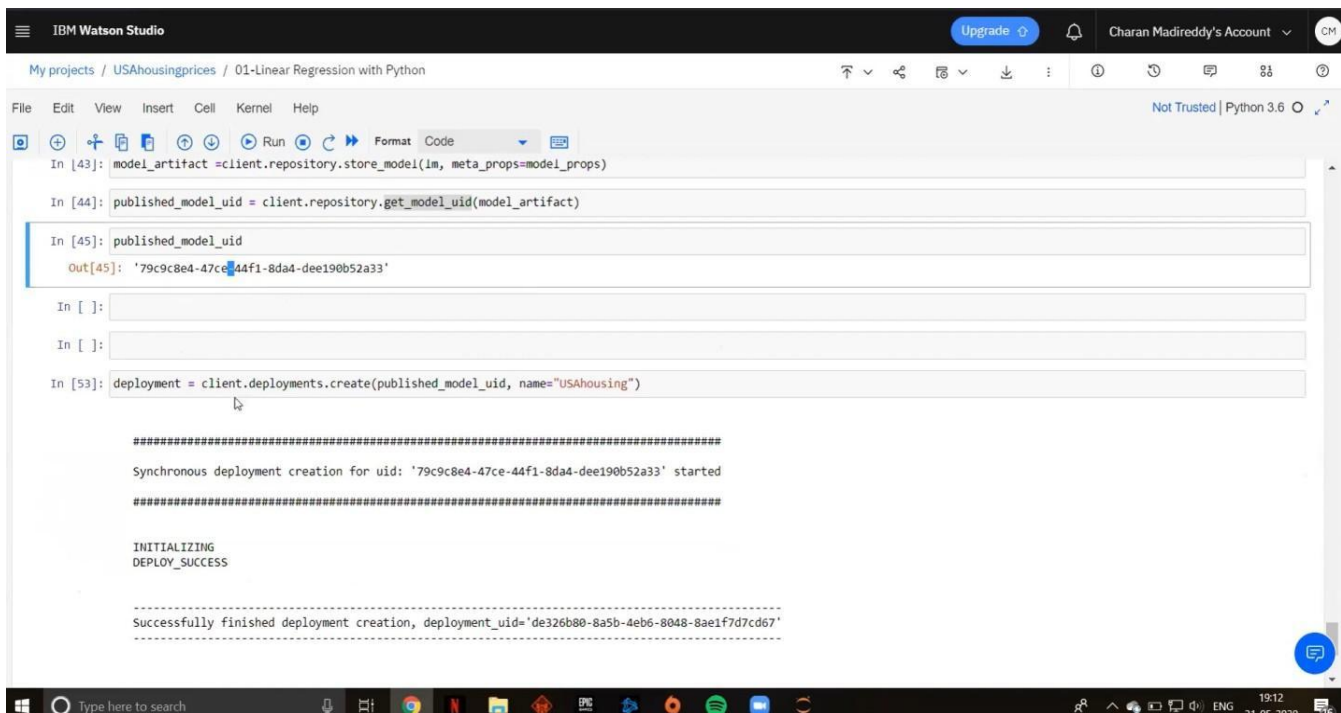
8. Publish and share:

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



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 breadcrumb navigation shows 'My projects / USAhousingprices / 01-Linear Regression with Python'. The main workspace contains a Jupyter Notebook with the following code and output:

```
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
#####

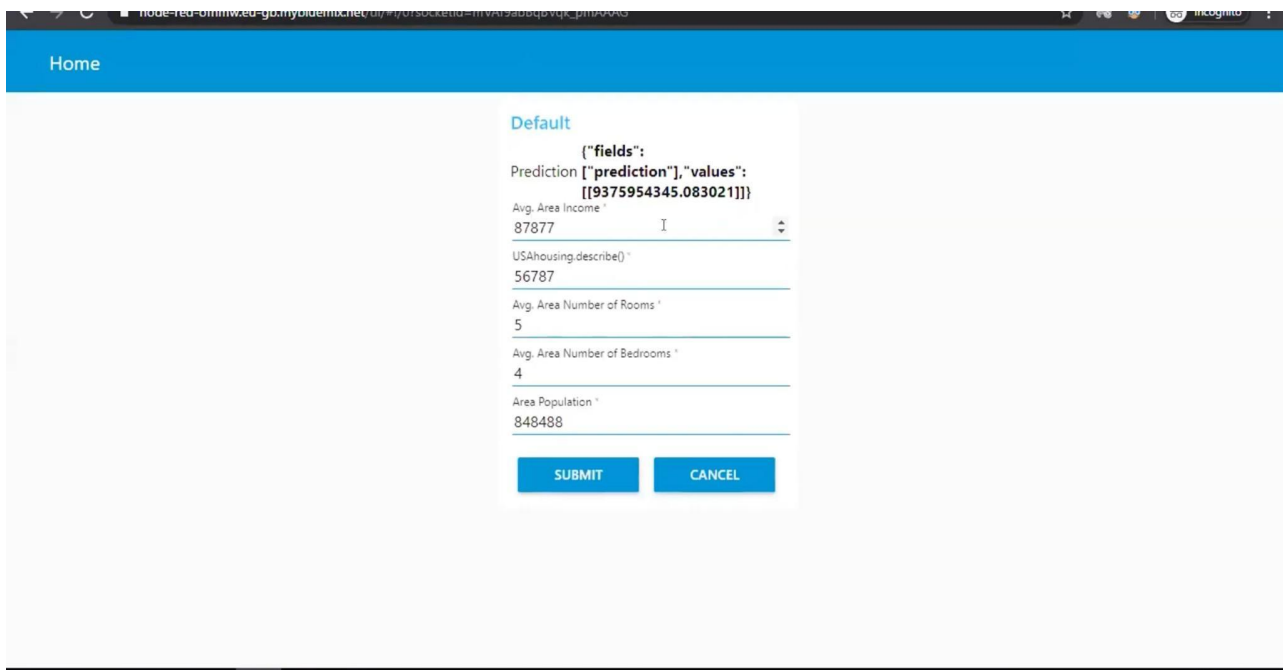
INITIALIZING
DEPLOY_SUCCESS

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

The interface also shows a top navigation bar with 'IBM Watson Studio', an 'Upgrade' button, and a user profile 'Charan Madireddy's Account'. A bottom status bar indicates 'Not Trusted | Python 3.6'.

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". The main content area is light gray. In the center, there is a white box titled "Default" containing a JSON-like structure for a prediction. The structure is as follows:

```
{
  "fields": {
    "prediction": "prediction",
    "values": {
      "prediction": 9375954345.083021
    }
  }
}
```

Below the JSON structure, there are five input fields with labels and values:

- Avg. Area Income: 87877
- USAhousing.describe(): 56787
- Avg. Area Number of Rooms: 5
- Avg. Area Number of Bedrooms: 4
- Area Population: 848488

At the bottom of the white box, there are two blue buttons: "SUBMIT" and "CANCEL".