**Machine Learning Model Deployment with IBM Cloud Watson Studio**

**BATCH MEMBERS**

**732721205015: Gobika . R**

**732721205017: Gomathi . M**

**732721205020: Hemamalini . S**

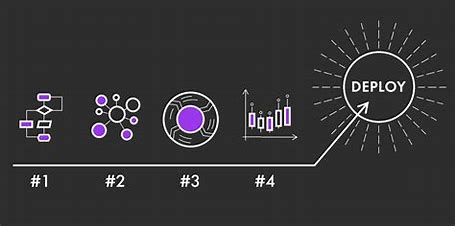
**732721205022: Janarthini . S**

**732721205023: Jayasri . S**

**Phase 4 Submission document**

**Project Title:** Model Deployment Machine Learning

**Topic:** Continue building the project by deploying the model and integrating it into applications. Deploy the trained model as a web service in IBM Cloud Watson Studio. Integrate the deployed model into applications using the provided API endpoint.



**Model Deployment Machine Learning**

**Version Control (Optional):**

Consider implementing version control for your deployed models. This allows you to keep track of different iterations and roll back to previous versions if needed.

**Set Up Authentication (Optional):**

Depending on your application's requirements, you may want to implement authentication and authorization mechanisms for accessing the API endpoint. This could involve using API keys, tokens, or other authentication methods supported by your deployment platform.

**Error Handling and Resilience:**

Implement robust error handling in your application code. This includes handling cases where the model service is unavailable, dealing with unexpected responses, and providing appropriate feedback to users.

**Logging and Monitoring:**

Integrate logging and monitoring solutions to keep track of API requests, responses, and any potential issues. This will help you identify and address problems quickly.

**Scaling and Performance Optimization:**

Depending on your application's requirements and usage patterns, you may need to consider scaling options for your deployed model. This could involve horizontal scaling (adding more instances) or vertical scaling (upgrading hardware).

**Compliance and Security:**

Ensure that your deployment complies with any legal, regulatory, or industry-specific requirements. This may include data privacy, security, and other compliance measures.

**Documentation:**

Provide clear and comprehensive documentation for developers who will be integrating with your API. Include information on the expected input format, API endpoints, and the format of the responses.

**Testing and Quality Assurance:**

Thoroughly test the integration of your deployed model in different scenarios to ensure it performs as expected. This includes testing with a variety of inputs to cover edge cases.

**Feedback Loop and Model Improvement:**

Establish a feedback loop to gather input from users and monitor the model's performance in real-world applications. Use this feedback to iterate and improve the model over time.

**Automated Deployment (Optional):**

If you anticipate frequent updates to your model, consider implementing automated deployment pipelines. This can streamline the process of deploying new versions.

**Continued Maintenance:**

Regularly monitor the performance of your deployed model and update it as necessary. Stay informed about advancements in the field of machine learning and consider retraining the model with new data periodically.

**Deploy the trained model as a web service in IBM Cloud Watson Studio.**

**Log in to IBM Cloud:**

Go to IBM Cloud and log in to your IBM Cloud account.

**Access Watson Studio:**

Navigate to Watson Studio from the IBM Cloud dashboard.

**Create a New Project:**

If you don't have an existing project, create a new one. This is where you'll manage your assets, including the deployed model.

**Add a Model Asset:**

Inside your project, go to the Assets tab and click on Add to project > Model. Select the appropriate option based on your model type and location (e.g., file upload, GitHub repository, etc.).

**Configure the Model:**

Depending on your model type, you may need to configure additional details like runtime environment, dependencies, and associated files.

**Train and Save the Model:**

If your model requires training, do so within Watson Studio. Save the trained model in a format compatible with deployment options available in Watson Studio.

**Deploy the Model:**

a. Click on the trained model asset in your project.

b. Select the Deployments tab and click on Add deployment.

c. Choose a deployment space (create one if needed) where the model will be hosted.

d. Select the deployment type based on your model type (e.g., Watson Machine Learning for machine learning models).

e. Follow the prompts to configure the deployment settings. This may include specifying the hardware configuration, scaling options, and other parameters.

**Start the Deployment:**

After configuring the deployment settings, initiate the deployment process. Watson Studio will now provision the necessary resources to host your model as a web service.

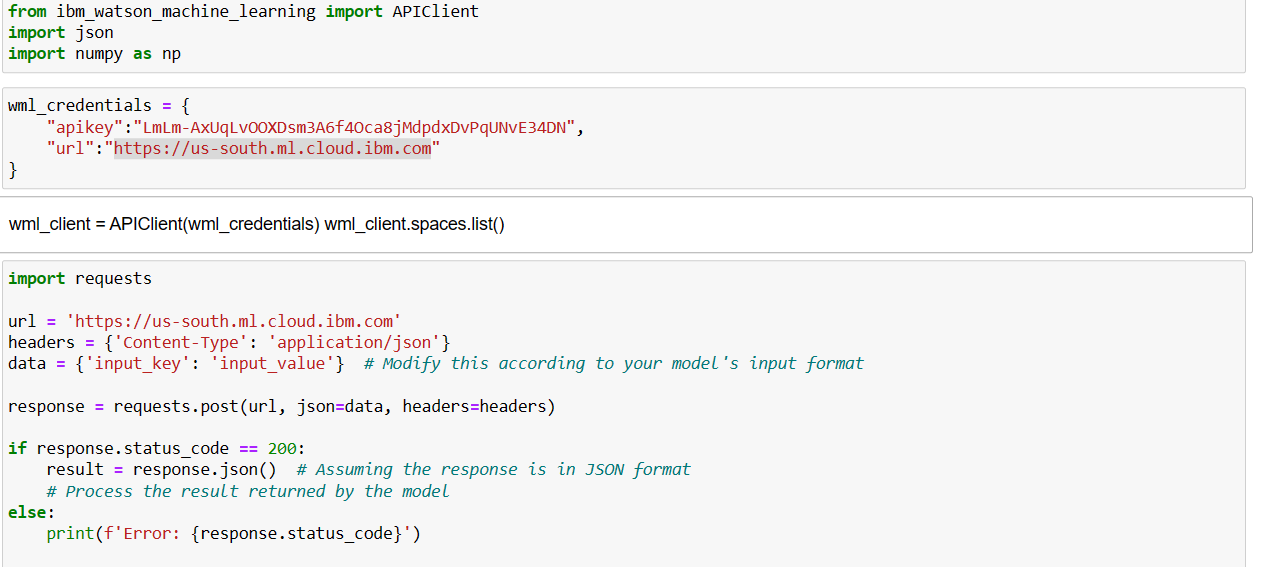
**Access the API Endpoint:**

Once the deployment is complete, you'll be provided with an API endpoint. This endpoint is the URL you'll use to make predictions with your deployed model.

**Integrate the Model into Applications:**

In your application code, use the API endpoint to send data to the deployed model for predictions. This usually involves making an HTTP POST request.

**IN [1]:**

****

The script will send a POST request to the specified API endpoint. If the request is successful (HTTP status code 200), it will assume the response is in JSON format and try to parse it. It will then print out the parsed result, which you would process according to the specific application.

Here's an example of what you might see as output if everything works correctly:

{

'output\_key': 'output\_value'

}

Of course, the actual output will depend on your model and the data you send. Keep in mind that the actual behavior will vary depending on the API endpoint and the behavior of the model deployed on that endpoint. This is just a template to get you started. If you want to see the specific output of this code, you'll need to replace 'YOUR\_API\_ENDPOINT' with the actual endpoint provided by Watson Studio and provide the correct input data. Then, when you run the code, it will interact with your specific deployed model.

**Handle Model Responses:**

Once you receive a response from the model, process it accordingly in your application. The format of the response will depend on how your model is configured to return predictions.

**Monitor and Manage the Deployment:**

Use the Watson Studio dashboard to monitor the performance of your deployed model. You can also manage and update the deployment as

**Choose a Programming Language:**

Decide which programming language you want to use to integrate the model. Common choices include Python, JavaScript (for web applications), Java, and others.

**Make HTTP Requests:**

Use the programming language's HTTP libraries or frameworks to make a POST request to the API endpoint.

**Process the Model's Response:**

If the request is successful, the result variable (assuming it's in JSON format) will contain the output from your model. Process this result according to your application's requirements.

**Error Handling:**

Implement proper error handling to deal with cases where the API request fails or returns an error status code. This is crucial for robust application behavior.

**Integrate with User Interface (if applicable):**

If your application has a user interface (e.g., a web page), you'll need to integrate the model prediction into the appropriate part of the interface.

**Testing:**

Thoroughly test the integration to ensure that the model works as expected within your application. Test with various inputs to cover different scenarios.

**Security Considerations:**

If applicable, consider implementing authentication and authorization mechanisms to secure access to the API endpoint.

**Deployment and Version Control (Optional):**

If you're deploying your application, ensure you have a process in place for deploying new versions and managing different iterations of your application.

**Logging and Monitoring (Optional):**

Implement logging and monitoring to keep track of API requests, responses, and any potential issues.

**Continued Maintenance:**

**Regularly monitor the performance of your application and model. Update the model or application as necessary to improve performance or add new features.** Decide which programming language you want to use to integrate the model. Common choices include Python, JavaScript (for web applications), Java, and others.

**CONCLUSION:**

* In the quest to build a house price prediction model, we have embarked on a critical journey that begins with loading and preprocessing the dataset. We have traversed through essential steps, starting with importing the necessary libraries to facilitate data manipulation and analysis.
* Understanding the data's structure, characteristics, and any potential issues through exploratory data analysis (EDA) is essential for informed decision-making.
* Data preprocessing emerged as a pivotal aspect of this process. It involves cleaning, transforming, and refining the dataset to ensure that it aligns with the requirements of machine learning algorithms.