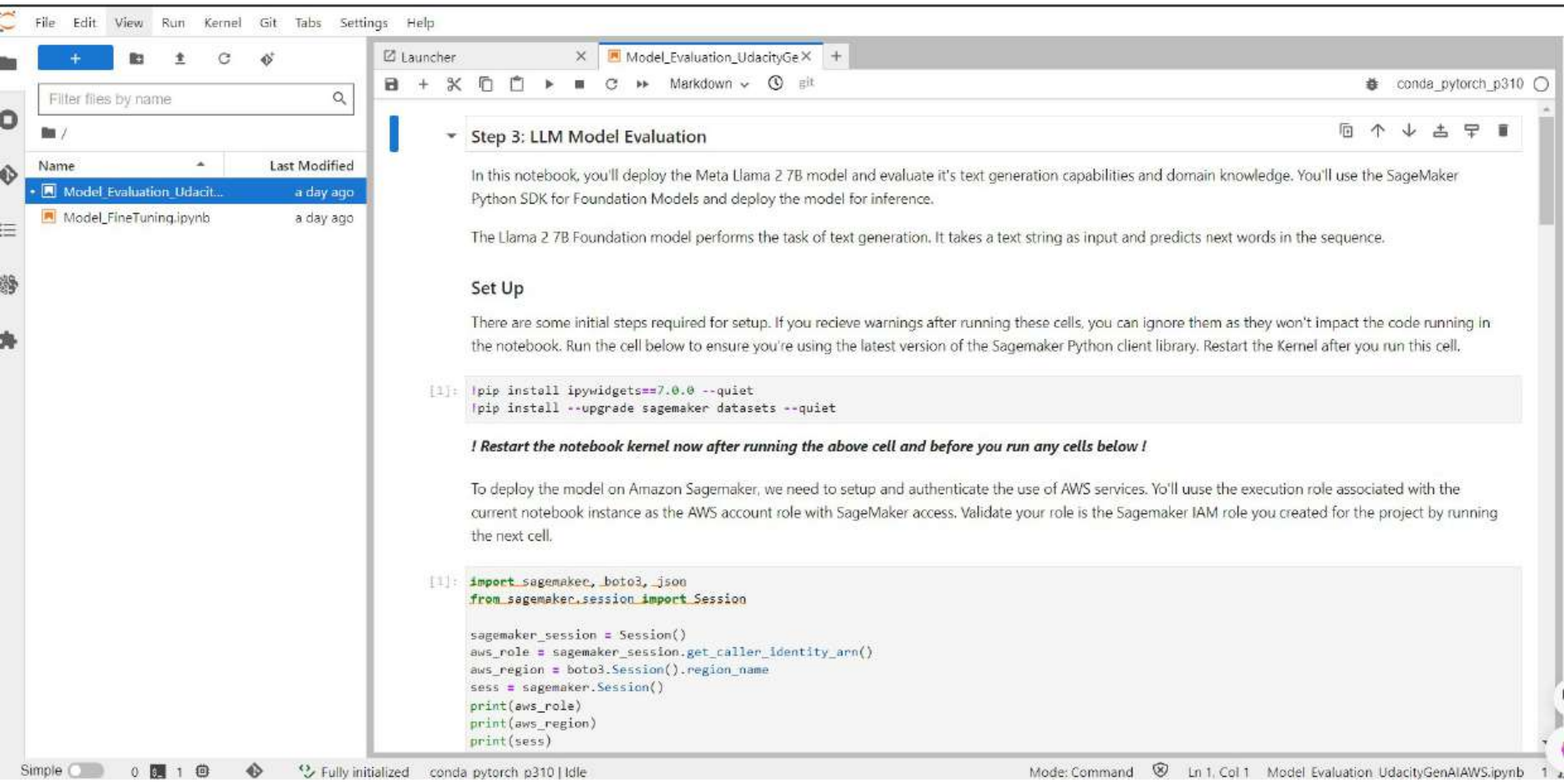


Step 3: LLM Model Evaluation

In this notebook, you'll deploy the Meta Llama 2 7B model and evaluate it's text generation capabilities and domain knowledge. You'll use the SageMaker Python SDK for Foundation Models and deploy the model for inference.

The Llama 2 7B Foundation model performs the task of text generation. It takes a text string as input and predicts next words in the sequence.

Set Up

There are some initial steps required for setup. If you recieve warnings after running these cells, you can ignore them as they won't impact the code running in the notebook. Run the cell below to ensure you're using the latest version of the Sagemaker Python client library. Restart the Kernel after you run this cell.

[1]:

```
!pip install ipywidgets==7.0.0 --quiet
!pip install --upgrade sagemaker datasets --quiet
```

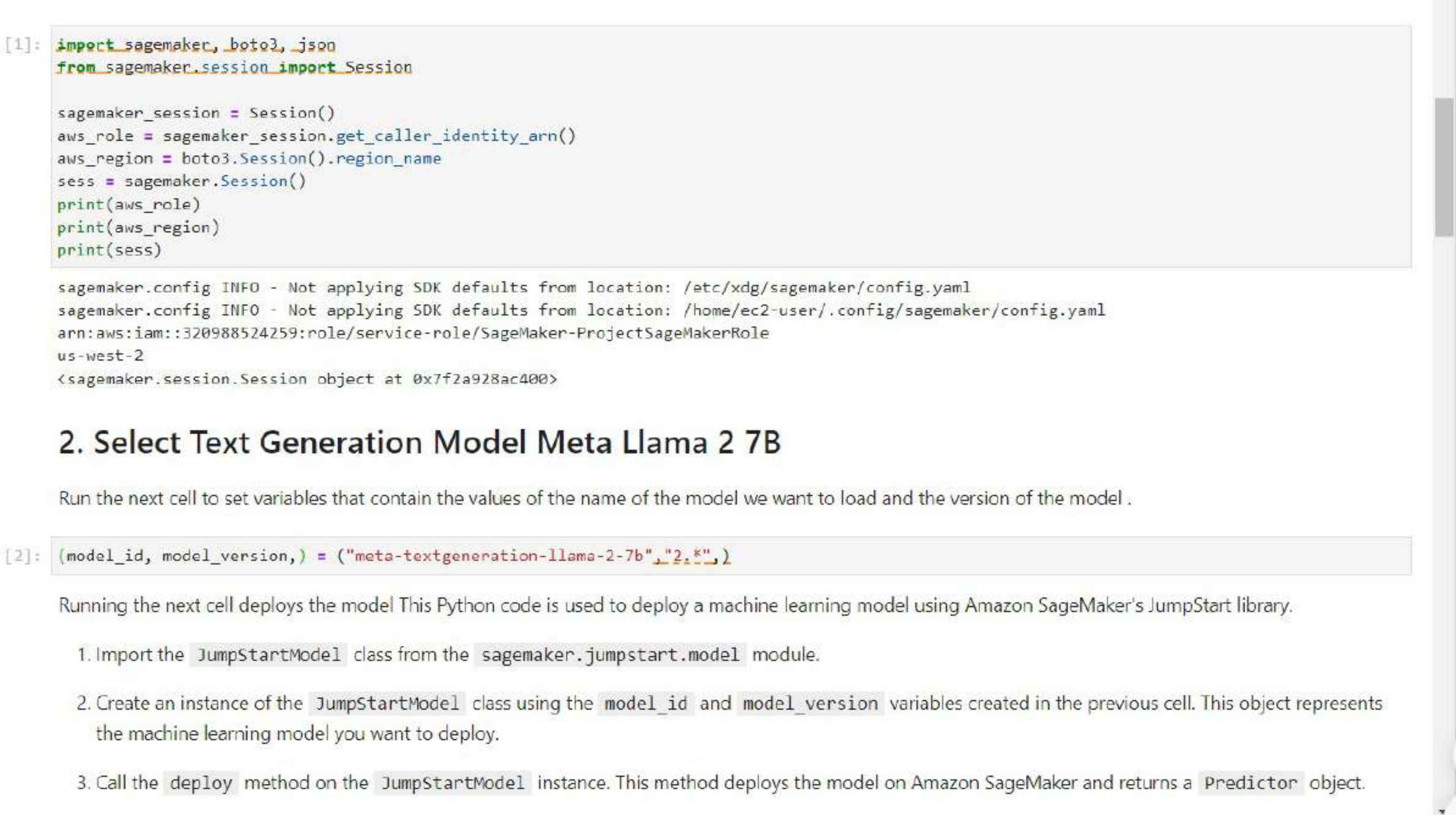
! Restart the notebook kernel now after running the above cell and before you run any cells below !

To deploy the model on Amazon Sagemaker, we need to setup and authenticate the use of AWS services. Yo'll uuse the execution role associated with the current notebook instance as the AWS account role with SageMaker access. Validate your role is the Sagemaker IAM role you created for the project by running the next cell.

[1]:

```
import sagemaker, boto3, json
from sagemaker.session import Session

sagemaker_session = Session()
aws_role = sagemaker_session.get_caller_identity_arn()
aws_region = boto3.Session().region_name
sess = sagemaker.Session()
print(aws_role)
print(aws_region)
print(sess)
```



```
[1]: import sagemaker, boto3, json
     from sagemaker.session import Session

     sagemaker_session = Session()
     aws_role = sagemaker_session.get_caller_identity_arn()
     aws_region = boto3.Session().region_name
     sess = sagemaker.Session()
     print(aws_role)
     print(aws_region)
     print(sess)
```

```
sagemaker.config INFO - Not applying SDK defaults from location: /etc/xdg/sagemaker/config.yaml
sagemaker.config INFO - Not applying SDK defaults from location: /home/ec2-user/.config/sagemaker/config.yaml
arn:aws:iam::320988524259:role/service-role/SageMaker-ProjectSageMakerRole
us-west-2
<sagemaker.session.Session object at 0x7f2a928ac400>
```

2. Select Text Generation Model Meta Llama 2 7B

Run the next cell to set variables that contain the values of the name of the model we want to load and the version of the model .

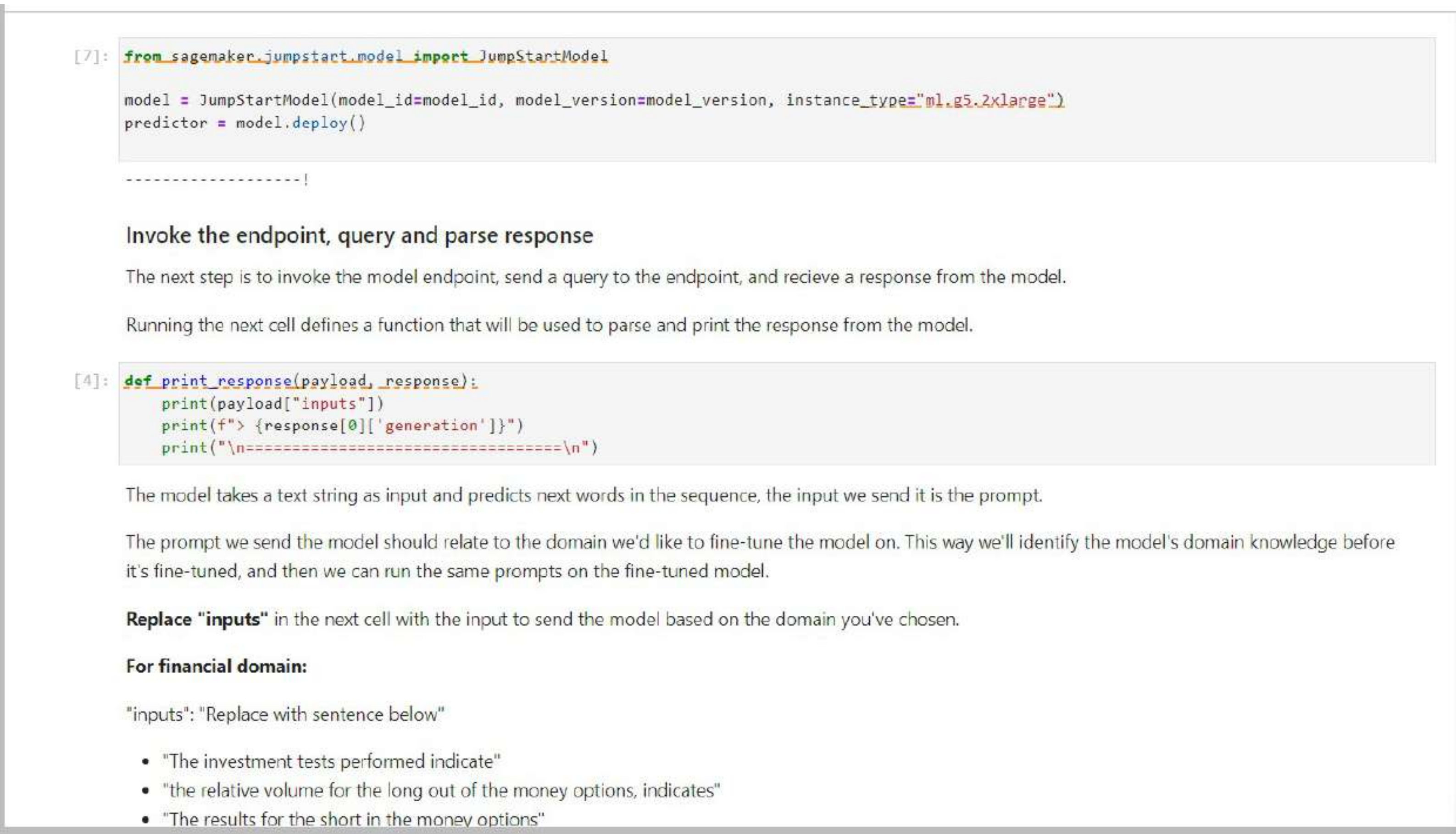
```
[2]: (model_id, model_version,) = ("meta-textgeneration-llama-2-7b","2.*",)
```

Running the next cell deploys the model This Python code is used to deploy a machine learning model using Amazon SageMaker's JumpStart library.

1. Import the `JumpStartModel` class from the `sagemaker.jumpstart.model` module.

2. Create an instance of the `JumpStartModel` class using the `model_id` and `model_version` variables created in the previous cell. This object represents the machine learning model you want to deploy.

3. Call the `deploy` method on the `JumpStartModel` instance. This method deploys the model on Amazon SageMaker and returns a `Predictor` object.



```
[7]: from sagemaker.jumpstart.model import JumpStartModel

     model = JumpStartModel(model_id=model_id, model_version=model_version, instance_type="ml.g5.2xlarge")
     predictor = model.deploy()
```

```
-------------------!
```

Invoke the endpoint, query and parse response

The next step is to invoke the model endpoint, send a query to the endpoint, and recieve a response from the model.

Running the next cell defines a function that will be used to parse and print the response from the model.

```
[4]: def print_response(payload, response):
         print(payload["inputs"])
         print(f"> {response[0]['generation']}")
         print("\n==================================\n")
```

The model takes a text string as input and predicts next words in the sequence, the input we send it is the prompt.

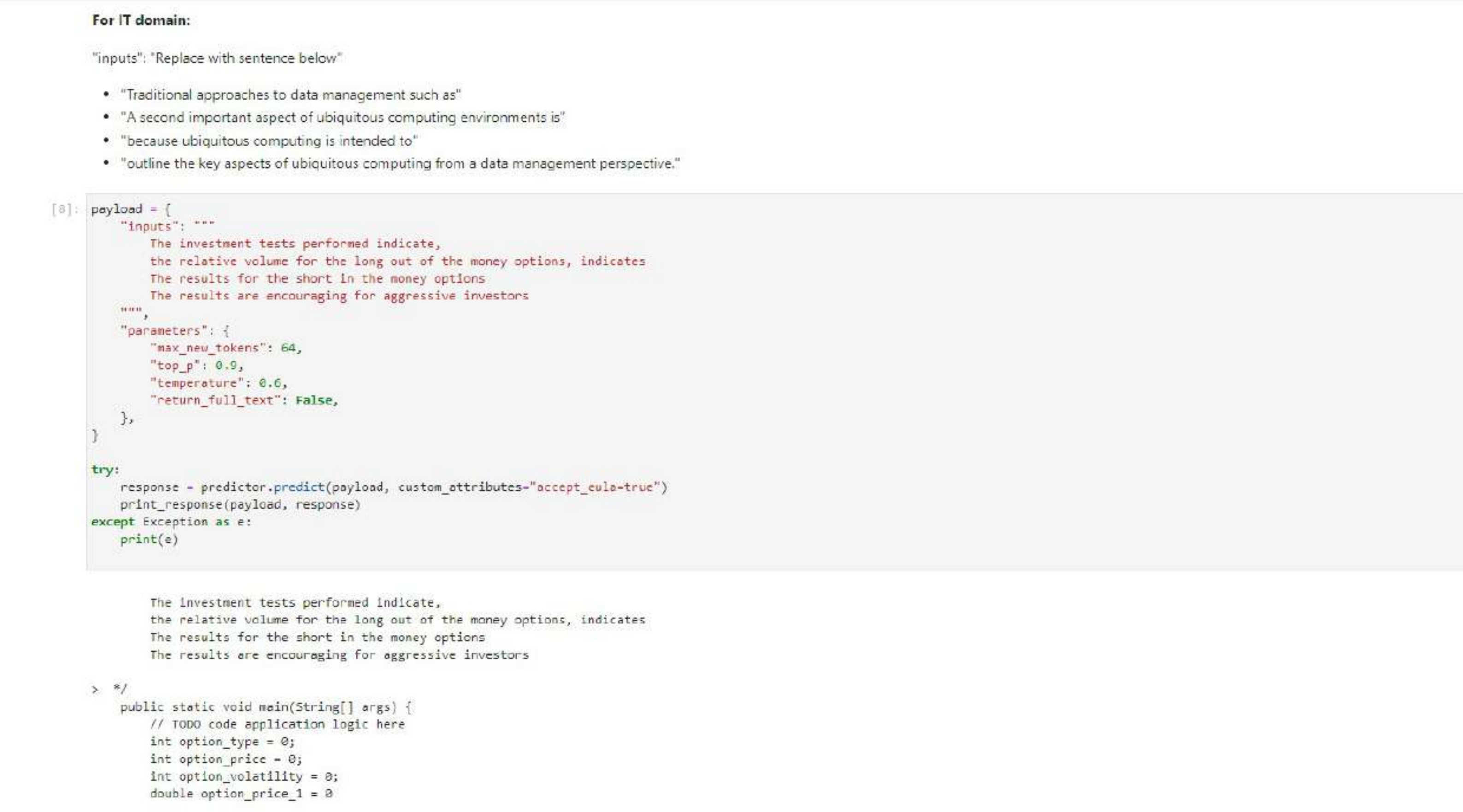
The prompt we send the model should relate to the domain we'd like to fine-tune the model on. This way we'll identify the model's domain knowledge before it's fine-tuned, and then we can run the same prompts on the fine-tuned model.

Replace "inputs" in the next cell with the input to send the model based on the domain you've chosen.

For financial domain:

"inputs": "Replace with sentence below"

- "The investment tests performed indicate"
- "the relative volume for the long out of the money options, indicates"
- "The results for the short in the money options"



For IT domain:

"inputs": "Replace with sentence below"

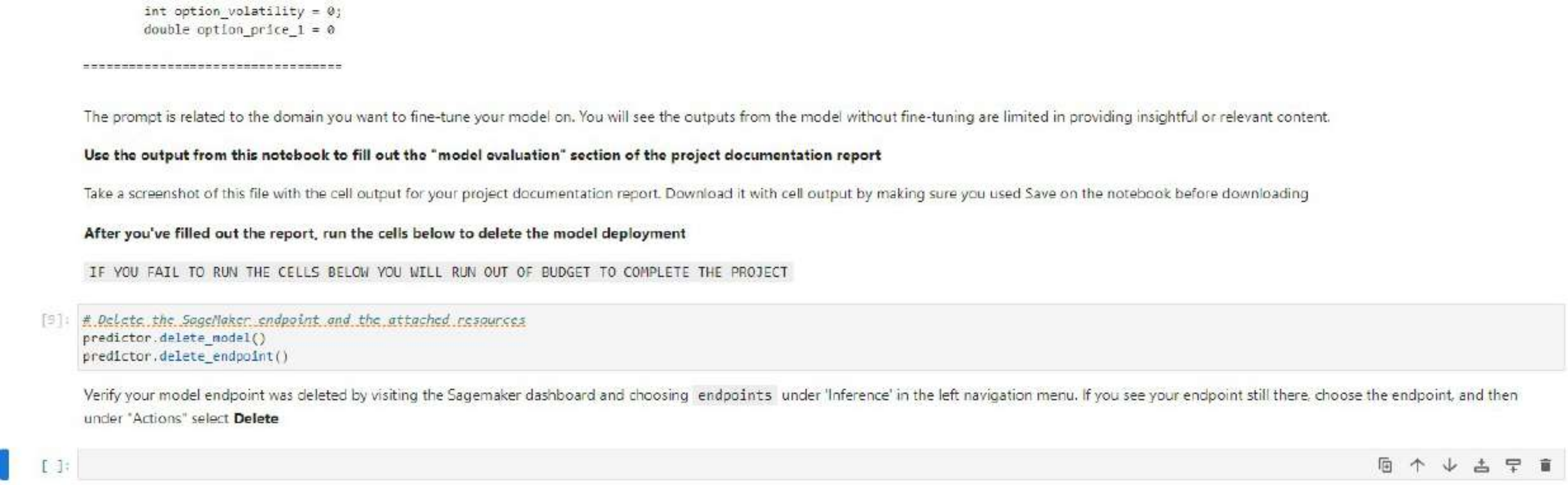
- "Traditional approaches to data management such as"
- "A second important aspect of ubiquitous computing environments is"
- "because ubiquitous computing is intended to"
- "outline the key aspects of ubiquitous computing from a data management perspective."

```
[8]: payload = {
         "inputs": """
             The investment tests performed indicate,
             the relative volume for the long out of the money options, indicates
             The results for the short in the money options
             The results are encouraging for aggressive investors
         """,
         "parameters": {
             "max_new_tokens": 64,
             "top_p": 0.9,
             "temperature": 0.6,
             "return_full_text": False,
         },
     }

     try:
         response = predictor.predict(payload, custom_attributes="accept_eula=true")
         print_response(payload, response)
     except Exception as e:
         print(e)
```

```
        The investment tests performed indicate,
        the relative volume for the long out of the money options, indicates
        The results for the short in the money options
        The results are encouraging for aggressive investors

>  */
    public static void main(String[] args) {
        // TODO code application logic here
        int option_type = 0;
        int option_price = 0;
        int option_volatility = 0;
        double option_price_1 = 0
```



```
        int option_volatility = 0;
        double option_price_1 = 0

==================================
```

The prompt is related to the domain you want to fine-tune your model on. You will see the outputs from the model without fine-tuning are limited in providing insightful or relevant content.

Use the output from this notebook to fill out the "model evaluation" section of the project documentation report

Take a screenshot of this file with the cell output for your project documentation report. Download it with cell output by making sure you used Save on the notebook before downloading

After you've filled out the report, run the cells below to delete the model deployment

`IF YOU FAIL TO RUN THE CELLS BELOW YOU WILL RUN OUT OF BUDGET TO COMPLETE THE PROJECT`

```
[9]: # Delete the SageMaker endpoint and the attached resources
     predictor.delete_model()
     predictor.delete_endpoint()
```

Verify your model endpoint was deleted by visiting the Sagemaker dashboard and choosing `endpoints` under 'Inference' in the left navigation menu. If you see your endpoint still there, choose the endpoint, and then under "Actions" select **Delete**

```
[ ]:
```