**Serverless Machine Learning Model Deployment with AWS SageMaker**

### A Course Project Report Submitted in partial fulfillment of the course requirements for the award of grades in the subject of

**CLOUD BASED AIML SPECIALITY (22SDCS07A)**

by

#### C SREE HASINI 2210030192

### Under the esteemed guidance of

##### Ms. P. Sree Lakshmi

Assistant Professor,

Department of Computer Science and Engineering



##### DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

##### K L Deemed to be UNIVERSITY

*Aziznagar, Moinabad, Hyderabad, Telangana, Pincode: 500075*

April 2025

**K L Deemed to be UNIVERSITY**

##### DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

### Certificate

This is Certified that the project entitled **“**Serverless Machine Learning Model Deployment with AWS SageMaker**”** which is a Experimental work carried out by C SREE HASINI (2210030192), in partial fulfillment of the course requirements for the award of grades in the subject of **CLOUD BASED AIML SPECIALITY**, during the year **2024-2025**. The project has been approved as it satisfies the academic requirements.

##### Ms.P.Sree Lakshmi Dr. Arpita Gupta

##### Course Coordinator Head of the Department

##### Ms. P. Sree Lakshmi Course Instructor

**CONTENTS**

Page No.

1. [Introduction 4](#_TOC_250005)
2. [AWS Services Used as part of the project 5](#_TOC_250004)
3. Steps involved in solving project problem statement 7
4. [Stepwise Screenshots with brief description 8](#_TOC_250003)
5. [Learning Outcomes 14](#_TOC_250002)
6. [Conclusion 15](#_TOC_250001)
7. [References 16](#_TOC_250000)

# INTRODUCTION

The **Serverless Machine Learning Model Deployment with AWS SageMaker** is a scalable and efficient solution that leverages cloud computing and serverless architectures to deploy machine learning models for real-time inference [1]. By using AWS SageMaker’s managed services, the project eliminates the need to maintain underlying infrastructure, ensuring rapid deployment and easy scaling of machine learning models. This system enables businesses to perform predictions on incoming data, automate decision-making workflows, and deliver AI-driven services efficiently [6]. It provides low-latency inference [3], secure model invocation [4], and an intuitive deployment process, helping organizations integrate machine learning into their applications seamlessly [6].

Designed as a cloud-native solution, the deployment approach enhances operational agility by automating model hosting, scaling, and monitoring tasks [6]. The platform ensures high availability, allowing applications to request predictions on-demand, store inference results [3], and deliver faster customer-facing services [6]. By leveraging serverless services like AWS Lambda and API Gateway [1], the system maintains a cost-effective, highly scalable, and maintenance-free architecture, making it ideal for modern AI applications [1].

The serverless ML deployment removes the need for managing servers or clusters, ensuring lower operational overhead and improved scalability [6]. It securely invokes trained SageMaker Endpoints via AWS Lambda functions [1], manages API requests through Amazon API Gateway [2], and enforces access control using IAM roles [4]. Performance and invocation metrics are monitored with Amazon CloudWatch [5], enabling proactive maintenance and optimization of the deployed machine learning services.

## AWS Services Used as part of the project

The project utilizes several AWS services to build a serverless architecture:

1. AWS SageMaker:

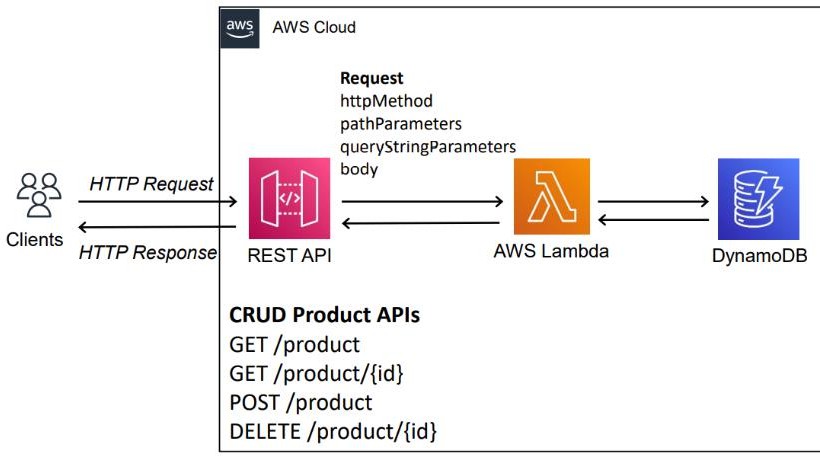
* Hosts the machine learning model for real-time inference [1].
* Manages model training, deployment, and endpoint scaling automatically [1].
* Provides a fully managed environment for hosting models without infrastructure management [1].



1. AWS Lambda:
   * Acts as an intermediary to invoke the SageMaker Endpoint for predictions [2]
   * Handles all API requests, including user interactions, data processing, and automated workflows [6].
   * Provides a serverless execution environment, scaling automatically based on demand [1].



1. Amazon API Gateway:
   * Acts as the entry point for all HTTP requests from the frontend [2].
   * Exposes the Lambda function as RESTful endpoints [2].
   * Manages CORS settings for cross-origin access from the web application [7].



1. AWS Identity and Access Management (IAM):
   * Used to create a role (CRMRole) with permissions for Lambda to interact with DynamoDB [4].
   * Ensures secure access to AWS resources [4].



These services work together to create a fully serverless application, minimizing operational overhead and ensuring scalability [1].

1. **Steps Involved in Solving the Project Problem Statement**

The project aimed to build a serverless machine learning model deployment architecture to enable efficient, scalable real-time inference. The following steps were taken to implement the solution:

1. **Model Development and Training**:

* Developed and trained a machine learning model (e.g., XGBoost classifier) using AWS SageMaker Notebooks [1].
* Preprocessed input data and tuned hyperparameters for optimal model accuracy [1].

1. **Deploying the SageMaker Endpoint**:

* Created and deployed a SageMaker Endpoint named "ml-endpoint" for real-time predictions [1].
* Configured the endpoint with appropriate instance types and auto-scaling settings [1].

1. **Creating the Lambda Function**:

* Developed a Lambda function named "InvokeMLFunction" to send requests to the SageMaker Endpoint [2].
* Implemented logic for input preprocessing, invoking the endpoint, and returning predictions [2].

1. **Configuring API Gateway**:

* Set up an API Gateway (MLAPI) to expose the Lambda function as a RESTful endpoint (e.g., /predict) [3].
* Enabled CORS for cross-origin requests to allow integration with web or mobile clients [7].
* Deployed the API to a "prod" stage, obtaining the Invoke URL (e.g., https://api-id.execute-api.us-east-1.amazonaws.com/prod) [3].

1. **Developing the Frontend (Optional)**:

* Created a simple frontend interface (index.html) for submitting prediction requests.
* Integrated Axios to make HTTP POST requests to the API Gateway endpoint [2].
* Included input validation and user feedback messages based on model predictions.

1. **Deployment and Packaging**:

* Packaged the Lambda function along with necessary configuration files into a zip archive (ml\_lambda.zip) [2].
* Uploaded the package to the Lambda console and updated the function settings [2].

1. **Debugging and Issue Resolution**:

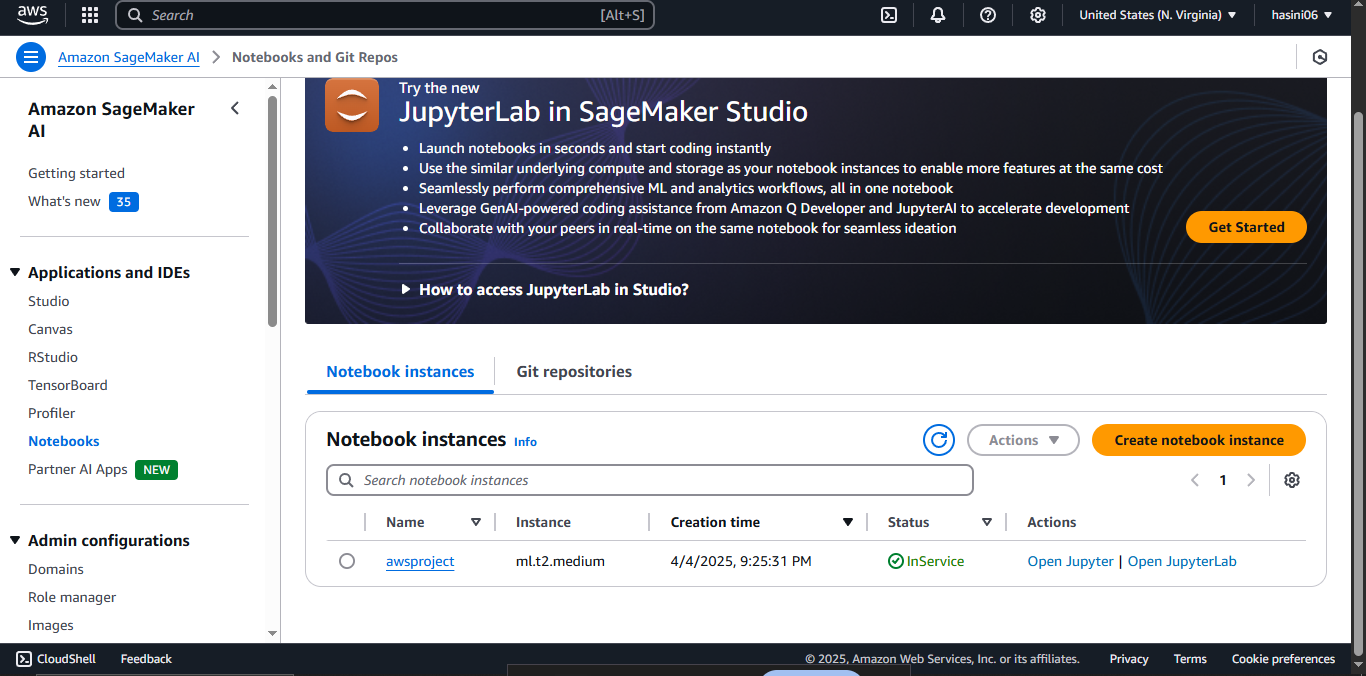
* Model Timeout Issues: Adjusted Lambda timeout settings and optimized endpoint invocation [5].
* CORS Problems: Properly configured API Gateway CORS settings to allow frontend communication [7].
* Serialization Issues: Ensured input and output data formats matched SageMaker expectations [5].

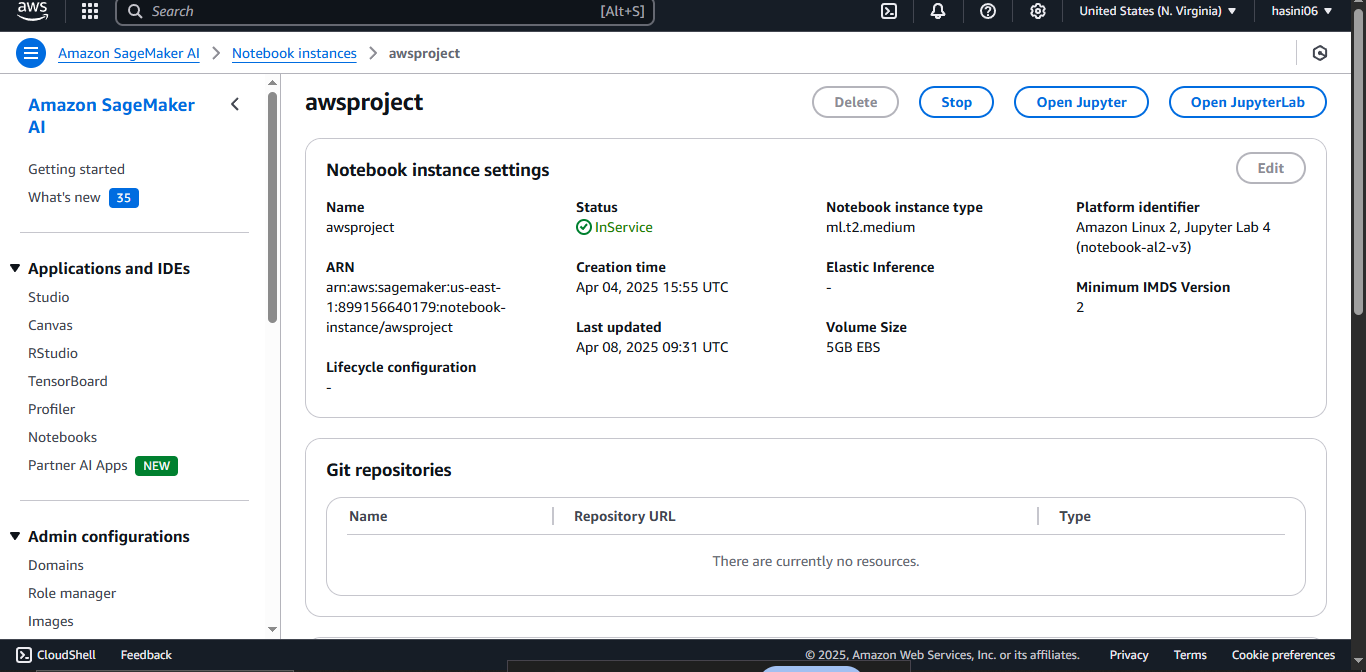
1. **Testing the Complete Workflow**:

* Sent test payloads via Postman and frontend to verify model predictions [2].
* Monitored SageMaker Endpoint and Lambda logs via Amazon CloudWatch to troubleshoot and optimize performance [5].

## Stepwise Screenshots with Brief Description

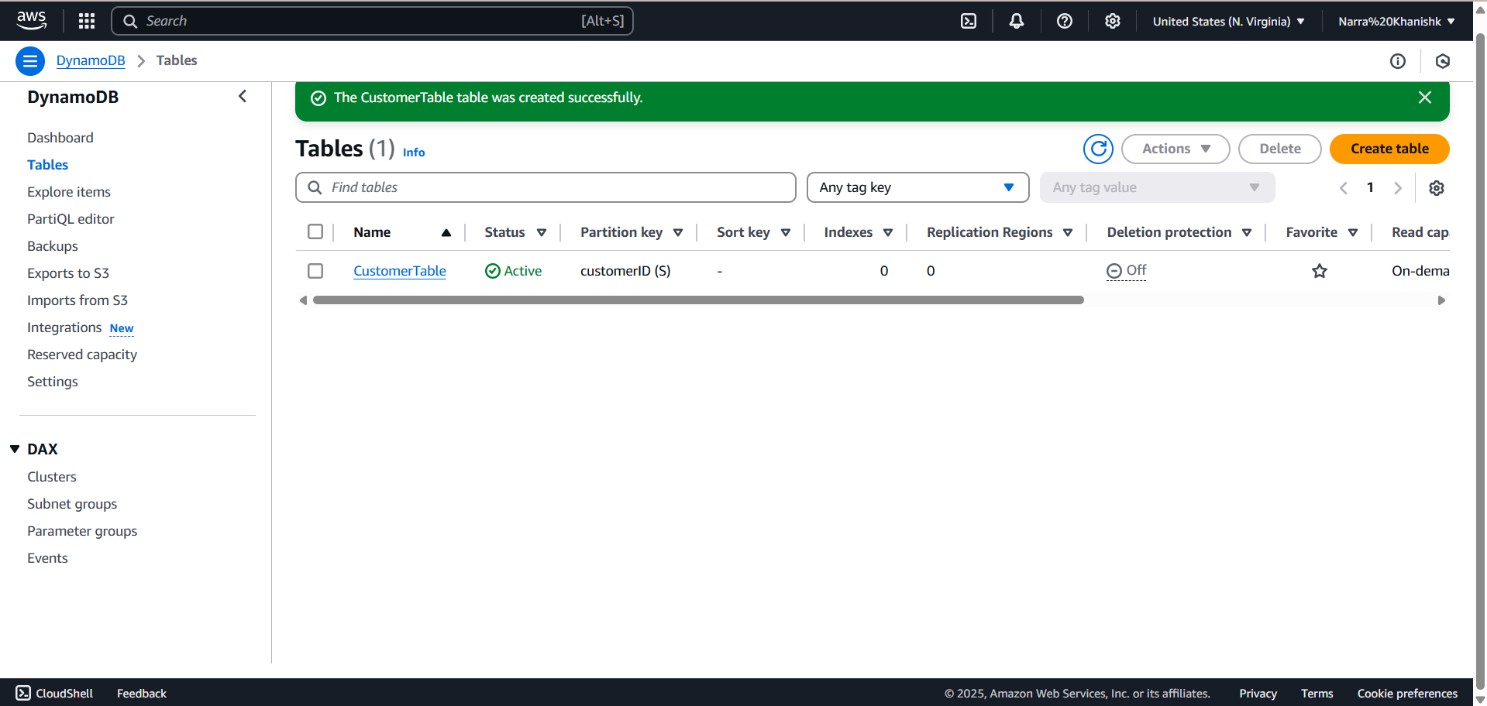
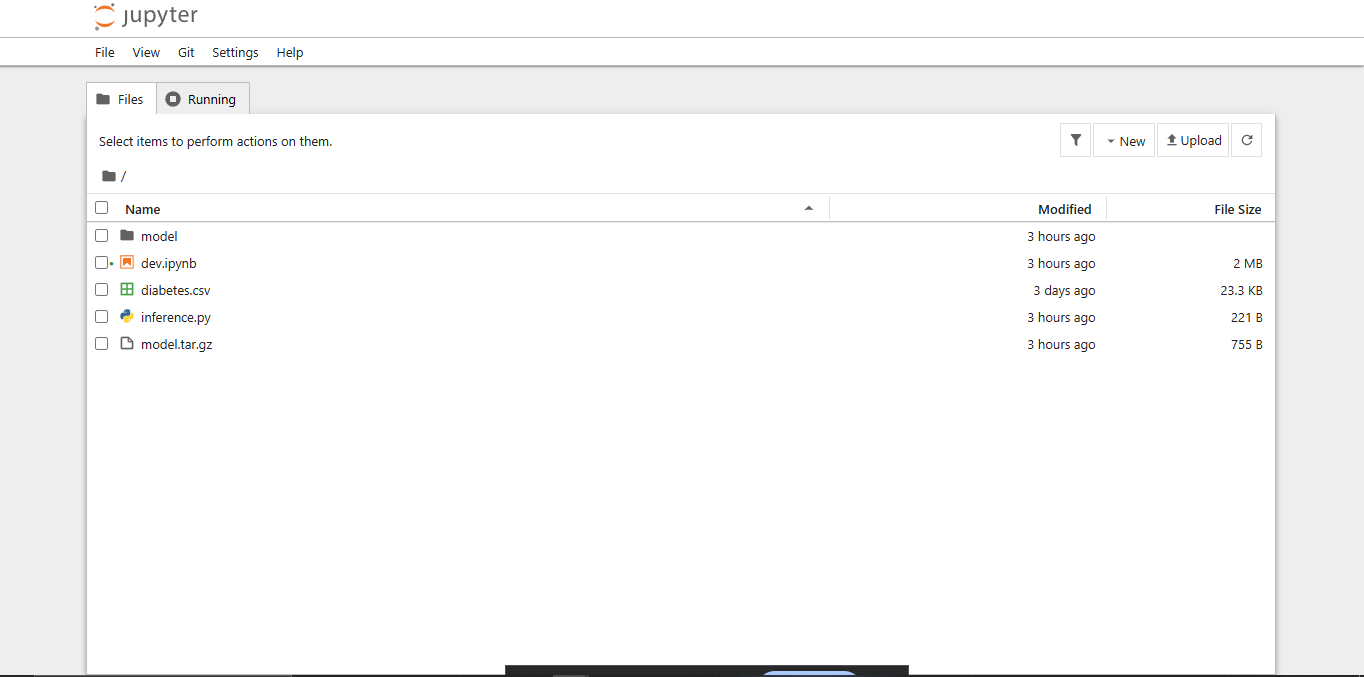
##### Screenshot 1: AWS SageMaker Notebook Setup

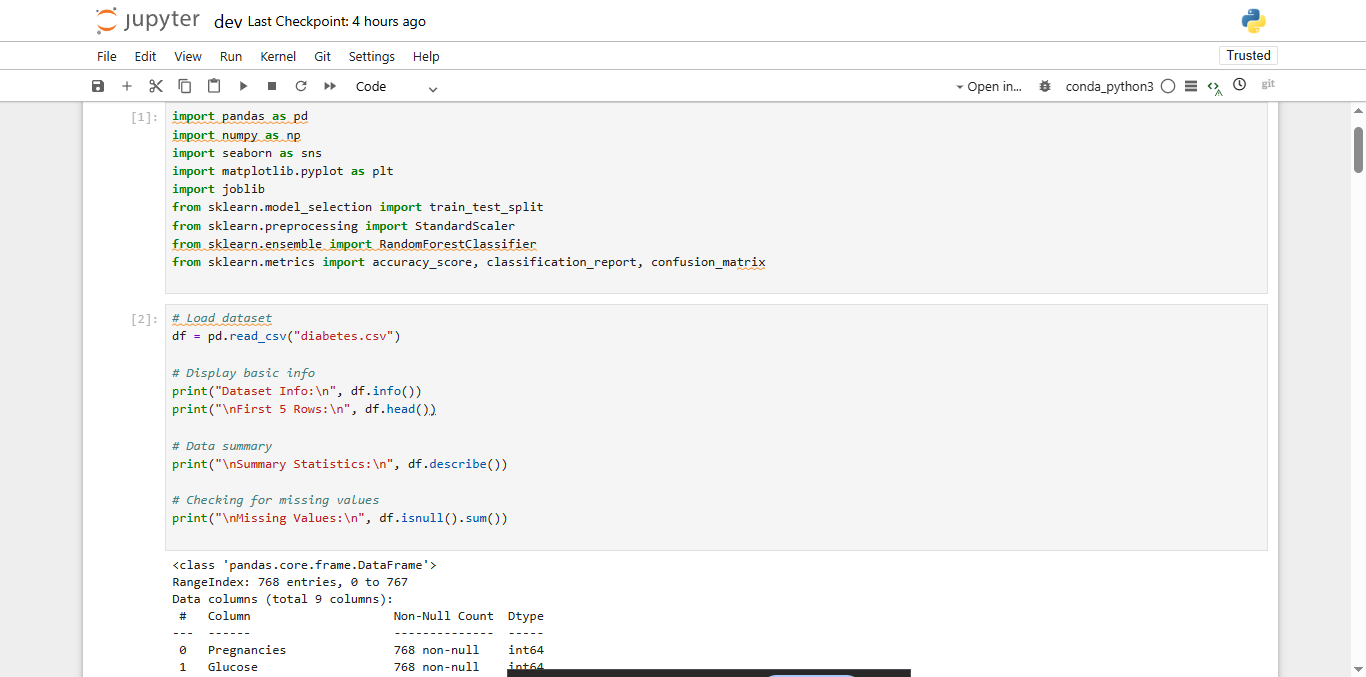
****

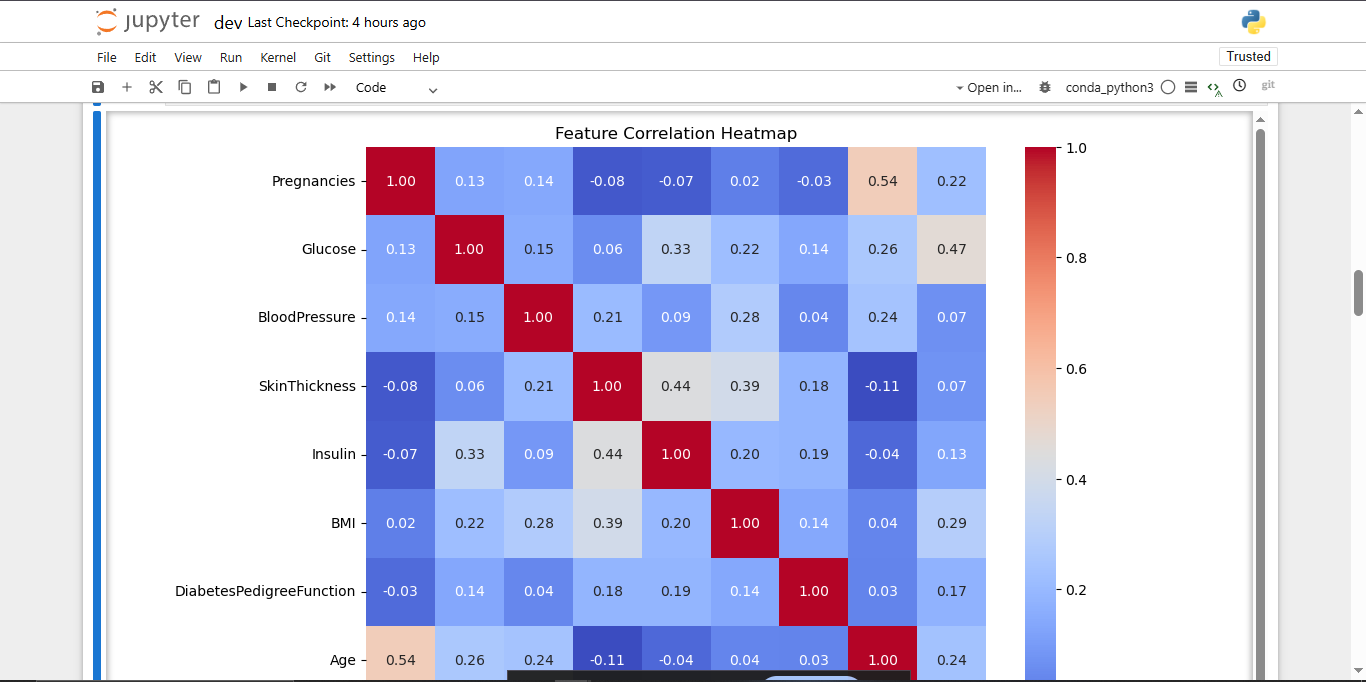
****

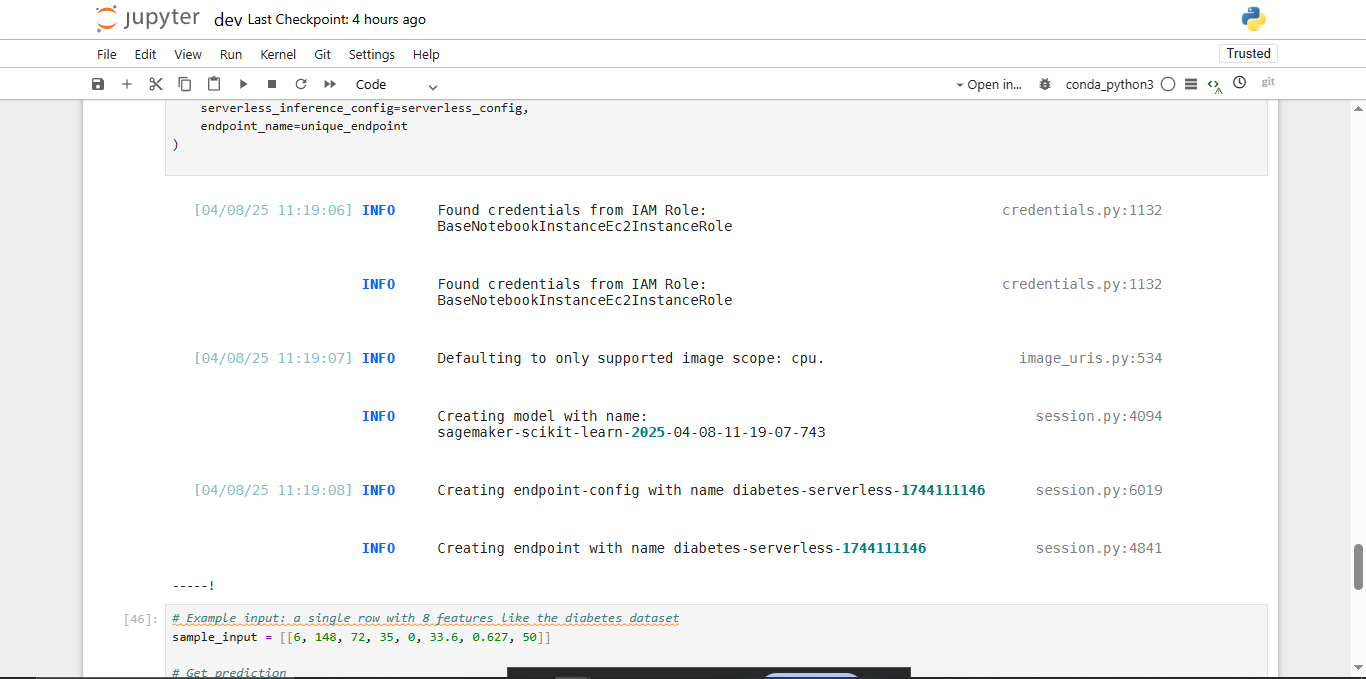
A Jupyter-based SageMaker notebook instance named awsproject is launched using an ml.t2.medium instance type [1]. It provides a cloud-hosted environment for developing, training, and testing machine learning models before deploying them to a SageMaker Endpoint for real-time inference as part of the serverless architecture [1].

##### Screenshot 2: Deploying the Python code in Jupyter

****

****

****

****

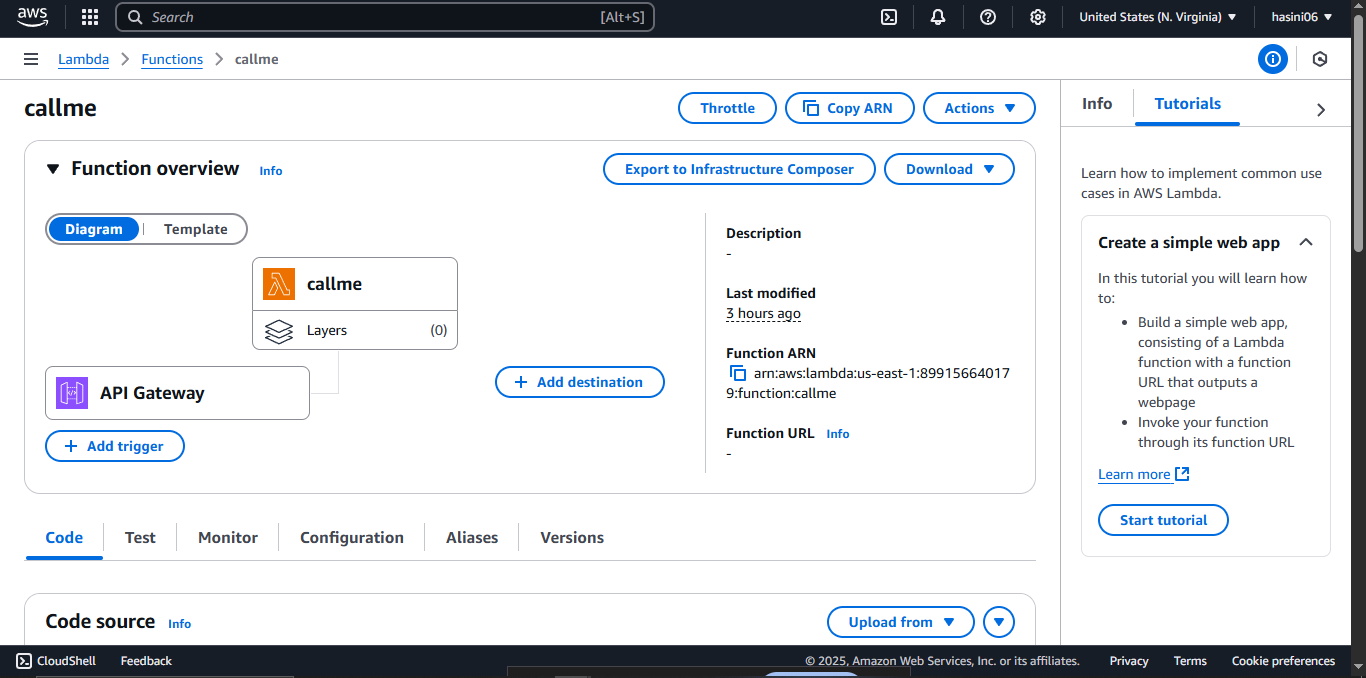
The visuals showcase key steps in deploying a serverless machine learning model using AWS SageMaker.

A Jupyter notebook instance named awsproject is used to train and test a RandomForestClassifier on the diabetes dataset using pandas and scikit-learn [1].

The notebook includes data preprocessing, correlation analysis, and model serialization for deployment.

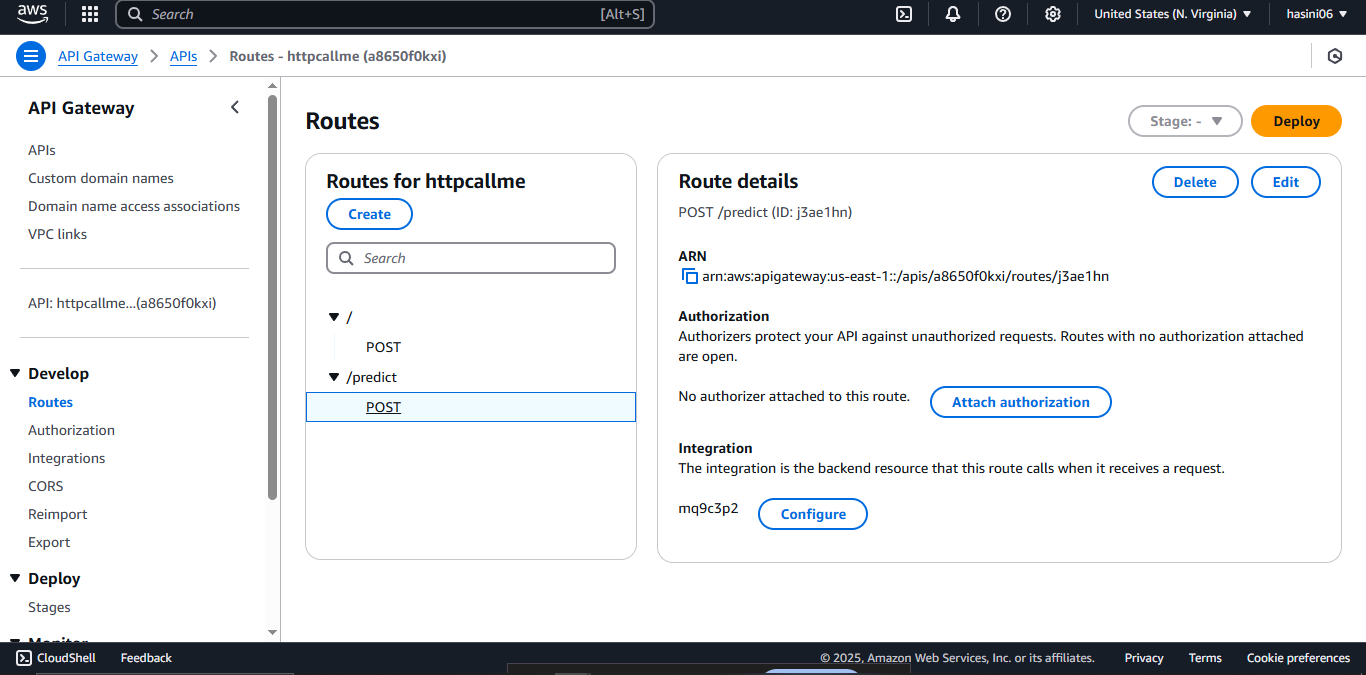
Logs confirm successful model packaging and SageMaker endpoint creation (diabetes-serverless-\*) for real-time inference [1][2].

##### Screenshot 3: Lambda Function Creation

****

This visual shows the creation of the CallmeFunction in the Lambda console [1]. The function is configured with the LambdaCallmeRole and set to use Python 3.13 as the runtime [1].

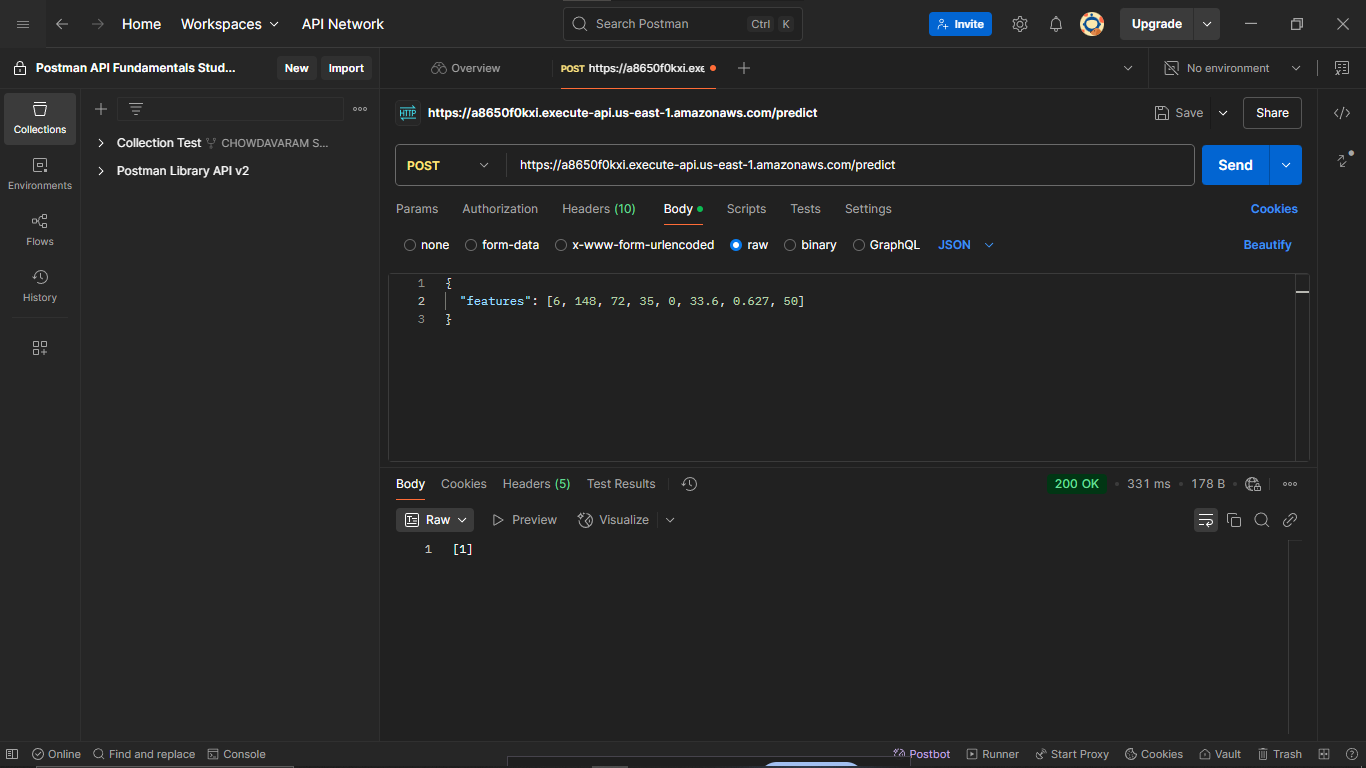
##### Screenshot 4: Attaching SageMaker Permissions to the API Gateway to execute in POSTMAN

****

A Jupyter notebook instance named awsproject is used to train and test a RandomForestClassifier on the diabetes dataset using pandas and scikit-learn [1]. The notebook includes data preprocessing, correlation analysis, and model serialization for deployment. Logs confirm successful model packaging and SageMaker endpoint creation (diabetes-serverless-\*) for real-time inference [1][2].

The image shows the configuration of a POST method in Amazon API Gateway for the /predict route. This route is connected to a backend Lambda function, which invokes the SageMaker endpoint for generating predictions. No authorizer is attached, making the endpoint publicly accessible unless restricted by other layers [2].

##### Screenshot 5: Final output

****

The image shows a POST method setup in API Gateway for the /predict route, integrated with a Lambda function to invoke the SageMaker endpoint [2].

This visual shows a successful Postman test of the /predict API with sample features. The 200 OK response confirms proper model invocation and result delivery [2].

## Learning Outcomes

1. Understanding Serverless Machine Learning Deployment:

Learned how to Deploy machine learning models using AWS SageMaker, lambda, and API Gateway in a fully serverless manner[1][2][3]. Understood the advantages of serverless ML, such as automatic scaling, reduced operational overhead, and cost optimization [1].

1. Working with AWS Services:

Gained hands-on experience with access to AWS SageMaker, lambda, and API Gateway,IAM [1][2][3][4][5].Learned how these services interact to deliver scalable and secure machine learning solutions[1][2].

1. IAM Role Configuration:

Understood the importance of IAM roles in securing serverless ML deployments [4]. Learned to create and attach policies to roles, ensuring least-privilege access for Lambda functions [4].

1. Debugging and Problem-Solving:

Developed skills in troubleshooting serverless applications using CloudWatch logs [5]. Successfully resolved issues such as endpoint timeout error, payload serialization problems, and CORS configuration errors between API Gateway and Lambda

1. API Gateway and Lambda

Integrated a frontend with a backend API using Axios [2]. Implemented CORS to allow cross-origin requests [7].

1. Best Practices in Development:

Learned the importance of error handling, logging, and data serialization in serverless applications [5]. Gained experience in packaging and deploying Lambda functions with dependencies [1].

## Conclusion

The Serverless Machine Learning Model Deployment project effectively demonstrates how AWS SageMaker, in conjunction with AWS Lambda, API Gateway, and other cloud-native tools, can be used to build a highly scalable, cost-efficient, and production-ready ML inference system. By adopting a serverless architecture, we eliminated the need for managing infrastructure, enabling automatic scaling and reduced operational complexity. The integration of AWS services allowed for seamless invocation of trained models via API endpoints, ensuring real-time predictions and minimal latency. Throughout the development process, we addressed key challenges such as model packaging, security permissions with IAM roles, and performance monitoring with CloudWatch, ultimately resulting in a robust and maintainable deployment pipeline.

This project underscored the flexibility and power of serverless solutions in deploying machine learning models, especially for applications that demand agility and scalability. The decoupled nature of serverless components made it easier to update and test models independently, while also simplifying version control and deployment workflows. Moving forward, enhancements such as implementing CI/CD pipelines for automated model updates, integrating data pipelines with SageMaker Pipelines, and deploying a web-based frontend via AWS Amplify or CloudFront can further improve usability and performance. Overall, the project highlights how serverless architectures can empower ML engineers and developers to bring intelligent applications to market faster and more efficiently.

## References

[1] AWS SageMaker Developer Guide:  
<https://docs.aws.amazon.com/sagemaker/latest/dg/whatis.html>

[2] AWS Lambda Developer Guide:  
<https://docs.aws.amazon.com/lambda/latest/dg/welcome.html>

[3] Amazon API Gateway Developer Guide:  
<https://docs.aws.amazon.com/apigateway/latest/developerguide/welcome.html>

[4] AWS IAM Documentation:  
<https://docs.aws.amazon.com/IAM/latest/UserGuide/introduction.html>

[5] Amazon CloudWatch Documentation:  
<https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/WhatIsCloudWatch.html>