# Configuration Manual

MSc Research Project
Programme Name

Forename Rajaram Jagadeeswaran
Student ID: x22239243

School of Computing
National College of Ireland

Supervisor: Jitendra Kumar Sharma

**National College of Ireland**

**MSc Project Submission Sheet**

**School of Computing**

| | |
|---|---|
| **Student Name:** | Rajaram Jagadeeswaran…… |
| **Student ID:** | X22239243……. |
| **Programme:** | MSc in Cloud Computing…… **Year:** …2024…….. |
| **Module:** | Research Project…….…… |
| **Lecturer:** | 16/09/2024…… |
| **Submission Due Date:** | ……. |
| **Project Title:** | Optimizing Healthcare Framework using Cognitive Computing Techniques in Cloud: A Study on Enhancing Diagnostic Accuracy and Decision-Making ……. |
| **Word Count:** | …400…… **Page Count:** …10……..…… |

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

**Signature:** …Rajaram Jagadeeswaran……

**Date:** …16/09/2024……

**PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST**

| | |
|---|---|
| Attach a completed copy of this sheet to each project (including multiple copies) | □ |
| **Attach a Moodle submission receipt of the online project submission,** to each project (including multiple copies). | □ |
| **You must ensure that you retain a HARD COPY of the project**, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer. | □ |

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

| **Office Use Only** | |
|---|---|
| Signature: | |
| Date: | |
| Penalty Applied (if applicable): | |

# Configuration Manual

Rajaram Jagadeeswaran
Student ID: x22239243

## 1 Prerequisite of AWS Account Setup

First make sure you have an active AWS account (private/ college). If you don't have one, sign up at AWS.

**IAM Roles & Permissions:** Set up required IAM Roles with actual permissions what is needed, which includes:

For adding new policy, need to use AWS Generate Policy. By adding valid ARN for any service and principal, you can be able to create new policies, follow this step throughout in all resources otherwise you can use the policy edit option to manually edit the policy for your need.

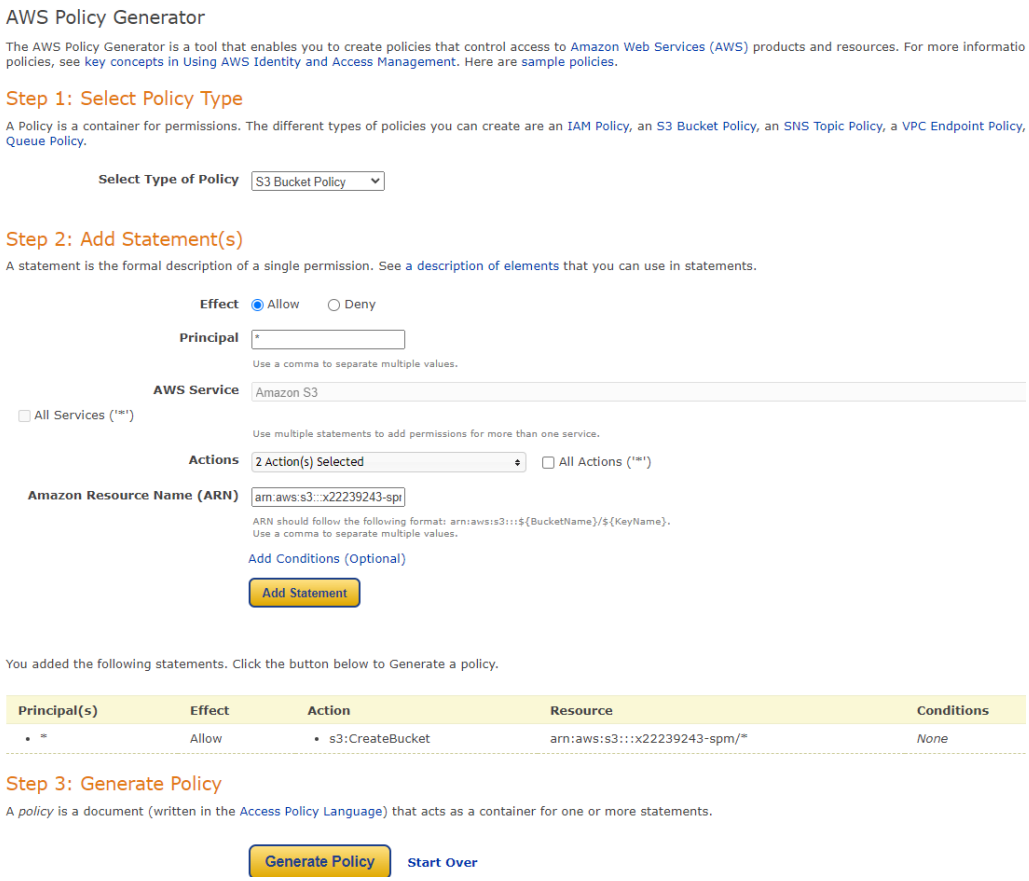

Figure 1. Generate policy

Once policy is generated you can copy it to add in services specifically.

## 1. SageMaker Execution Role:

This role should have permission to interact with S3 bucket wherever going to be used by SageMaker, Bedrock. If you are using multiple buckets, make sure to create execution roles for each S3 bucket.

- Below figure is the model of existing S3 bucket permission added via specific S3 bucket from permission tab.

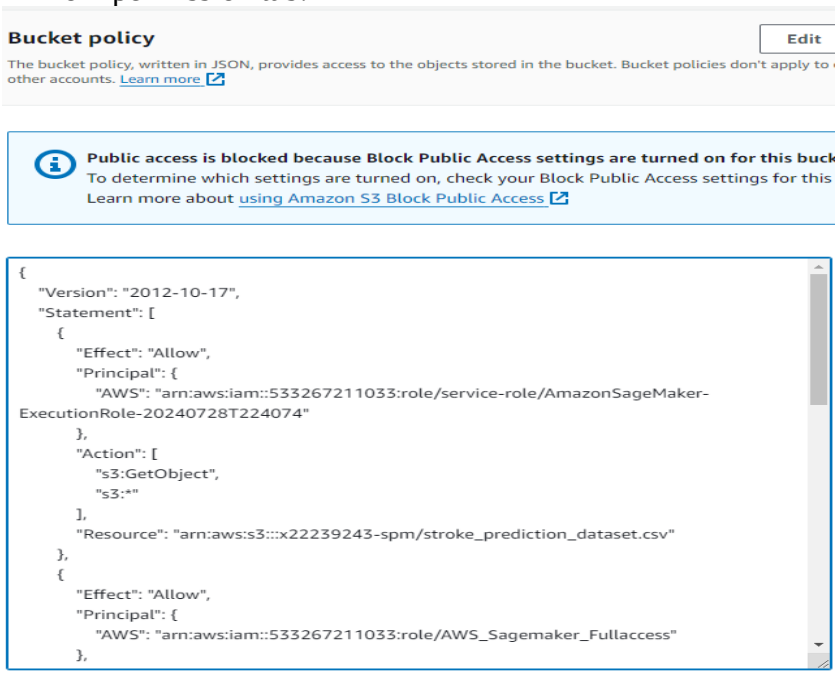


Figure 2. S3 Bucket Policy for utilizing in SageMaker

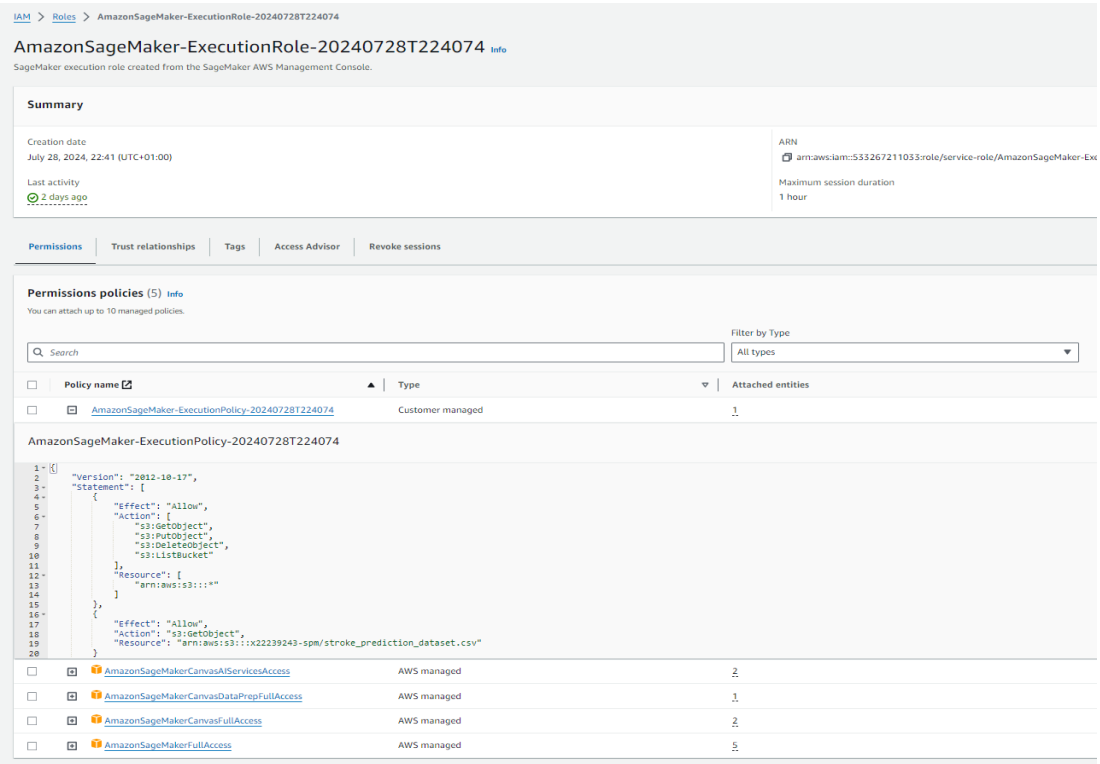


Figure 3. SageMaker execution Role Permission

## 2. Bedrock Role:

This role should have permissions to interact with Bedrock models and access the necessary AWS services.

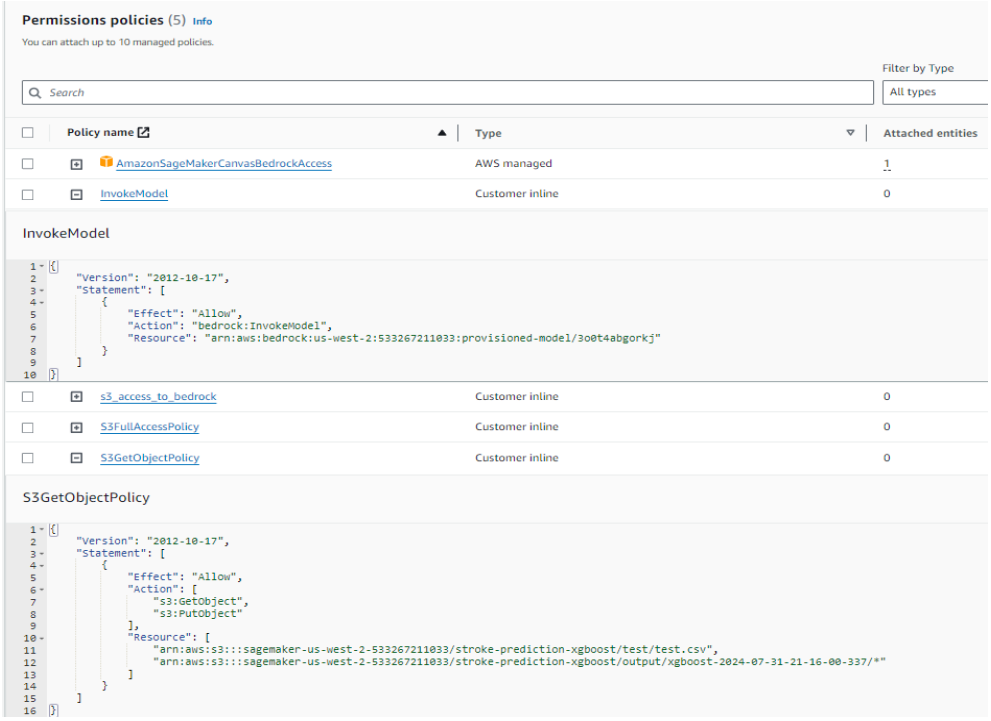


Figure 4. Bedrock execution Role Permission

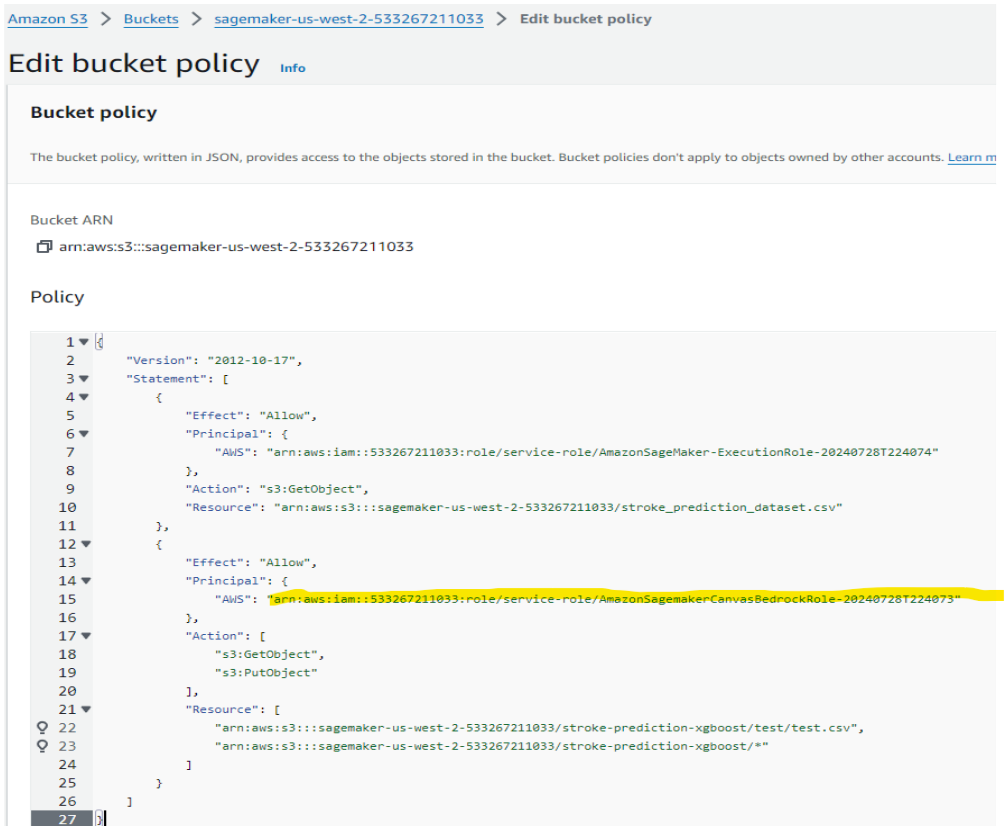


Figure 5. S3 Buckets Policy for Bedrock access

# 2 Environmental Setup

**Python:** Make sure you have installed python. If Python is not already installed on your system, you can download and install it from the official Python website. Follow the instruction according to your requirements. If you have already installed it, pls check with the below code to verify the latest version is installed.

```
PS C:\Users\rajar\OneDrive\Desktop\personal\stroke-prediction> python --version
PS C:\Users\rajar\OneDrive\Desktop\personal\stroke-prediction> python --version
>>
Python 3.11.5
```

Figure 6. Verify Python Version

**Libraries:** Make sure to install the required python libraries as shown in fig. 7.

```
import streamlit as st
import boto3
import pandas as pd
import numpy as np
from io import StringIO
import json
import time
```

Figure 7. Import Python libraries

**SageMaker:**

1. **Notebook Instance:** Jupyter Notebook setup is primary to create, train and deploy model, you can use local Jupyter software to work with. As shown in fig. 8 first need to create a new notebook.

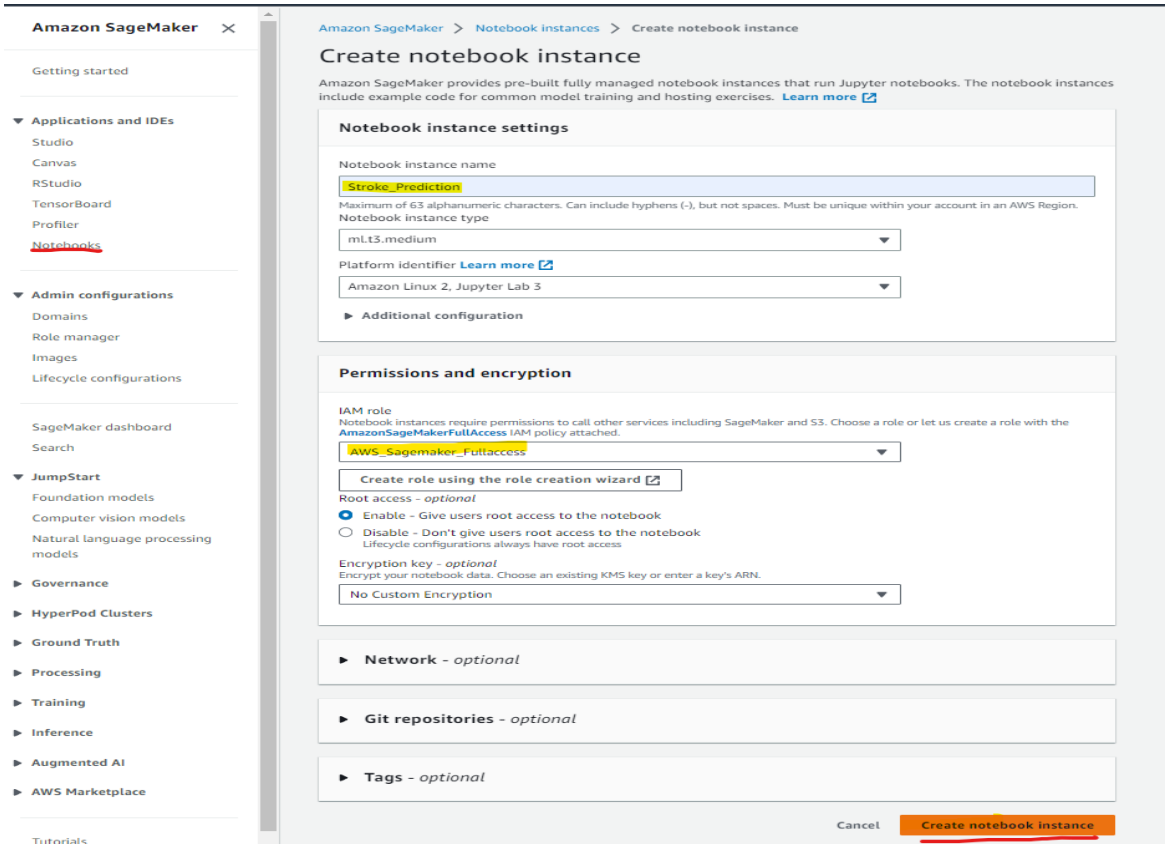


Figure 8. SageMaker Notebook Instance

For more user experience you can use SageMaker studio, Once Instance is created you can view the status as Running. Make sure to stop the service before closing every time.

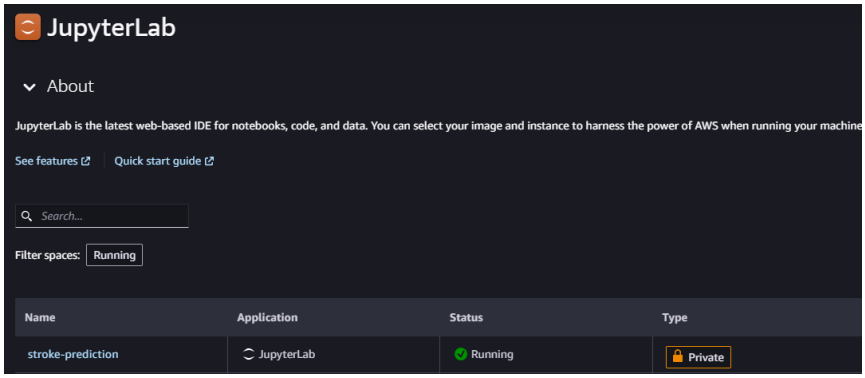

Figure 9. Notebook Instance – Inservice

2. **Dataset load from S3:** By specifying bucket name & region we can use objects from it.

```
region = boto3.Session().region_name
sagemaker_session = sagemaker.Session()
role = sagemaker.get_execution_role()
```

```
print(f"AWS Region: {region}")
print(f"RoleArn: {role}")
```

```
AWS Region: us-west-2
RoleArn: arn:aws:iam::533267211033:role/service-role/AmazonSageMaker-ExecutionRole-20240728T224074
```

```
# Specify your bucket name and file name
bucket_name = 'x22239243-spm'
file_key = 'stroke_prediction_dataset.csv'
```

```
# Read the CSV file from S3
s3 = boto3.client('s3')
csv_obj = s3.get_object(Bucket=bucket_name, Key=file_key)
body = csv_obj['Body']
csv_string = body.read().decode('utf-8')
```

Figure 10. Dataset load from S3 to be used entirely

3. **Model Training, Endpoint Deployment:** The model training job with creating and endpoint configurations are handled in code itself. Follow the below code,

```
# Define the XGBoost model
xgb = Estimator(
    image_uri=xgboost_container,
    role=role,
    instance_count=1,
    instance_type='ml.m4.xlarge',
    output_path=f"s3://{s3_bucket}/{s3_prefix}/output",
    sagemaker_session=sagemaker_session
)
```

```
# Set hyperparameters
xgb.set_hyperparameters(
    objective='binary:logistic',
    num_round=100,
    max_depth=5,
    eta=0.3,
    eval_metric='logloss'
)
```

```
# Define the data inputs
train_input = TrainingInput(s3_data=train_s3_path, content_type='csv')
test_input = TrainingInput(s3_data=test_s3_path, content_type='csv')
```

```
# Train the model
xgb.fit({'train': train_input, 'validation': test_input})
```

```
INFO:sagemaker:Creating training-job with name: xgboost-2024-07-31-21-16-00-337
2024-07-31 21:16:00 Starting - Starting the training job...
2024-07-31 21:16:22 Starting - Preparing the instances for training...
2024-07-31 21:16:49 Downloading - Downloading input data...
2024-07-31 21:17:20 Downloading - Downloading the training image.....
```

Figure 11. Model Training Job

Once The training job has been completed your model is created and ready to deploy.

```
2024-07-31 21:18:44 Completed - Training job completed
Training seconds: 115
Billable seconds: 115
```

```
[40]: # Deploy the model
      xgb_predictor = xgb.deploy(initial_instance_count=1, instance_type='ml.m4.xlarge')
```

```
INFO:sagemaker:Creating model with name: xgboost-2024-07-31-21-19-13-748
INFO:sagemaker:Creating endpoint-config with name xgboost-2024-07-31-21-19-13-748
INFO:sagemaker:Creating endpoint with name xgboost-2024-07-31-21-19-13-748
-------!
```

```
[41]: # Set the serializer and deserializer for the predictor
      xgb_predictor.serializer = CSVSerializer()
      xgb_predictor.deserializer = JSONDeserializer()
```

```
[37]: # Make predictions on the test set
      y_pred = []
      for index, row in X_test.iterrows():
          result = xgb_predictor.predict(row.values)
          y_pred.append(result)

      # Convert predictions to binary outcomes
      y_pred = np.round(y_pred)
```

Figure 12. Deploy model as Endpoint to test by invoking with prediction data.

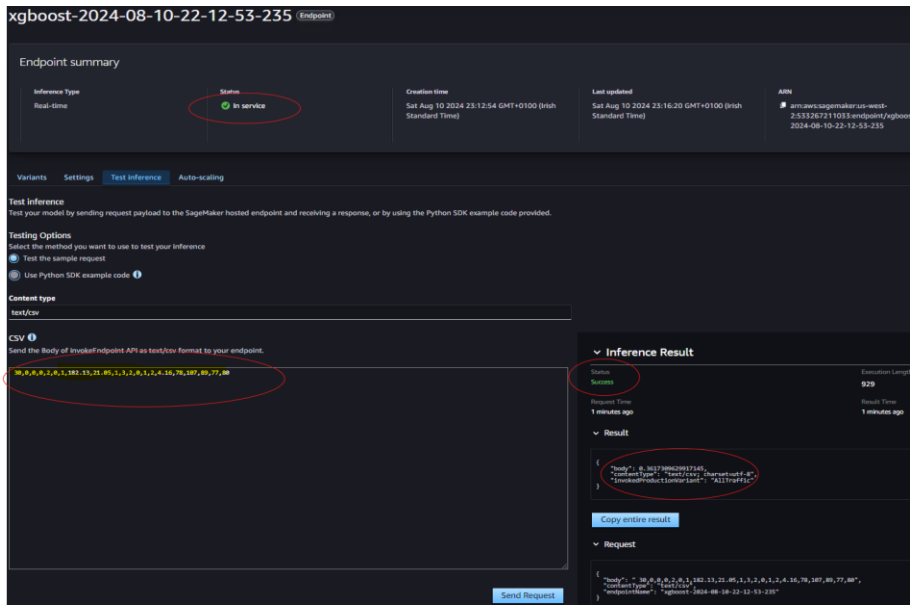


Figure 13. Evaluating the Endpoint by giving sample inputs

From the above figure, we can see the SageMaker endpoint evaluated successfully which has given the probability prediction score for risk of patient's stroke.

## Bedrock:

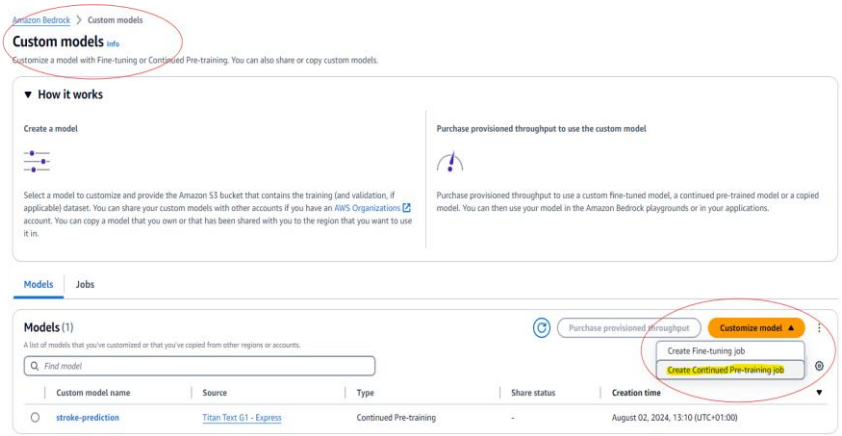


Figure 14. Bedrock custom model

As shown in fig. 14, Bedrock console has an option named custom model, here we need to select create continued pre-training job. Following this we need to select any available source model in specific region. If you have no available base model, then request for model access. Text Generation is the objective so Titan Text G1-Express need to be chosen. With the dataset stored from S3 need to select as input data. Note: Bedrock accepts only JSON format data, so the prior conversion is required from CSV file to JSON. That needs to be used as input data. The service role should be chosen appropriately as shown in fig. 15.

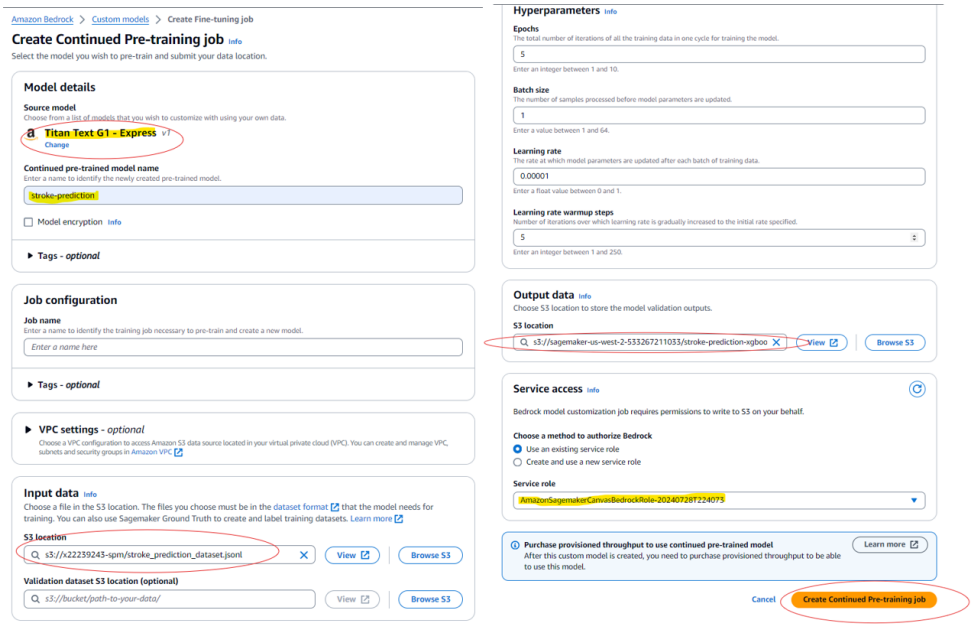


Figure 15. Bedrock custom model creation

Once the training job getting completed, we need to purchase provisioned throughput as deploy and used for evaluating the custom model. As shown in fig. 16

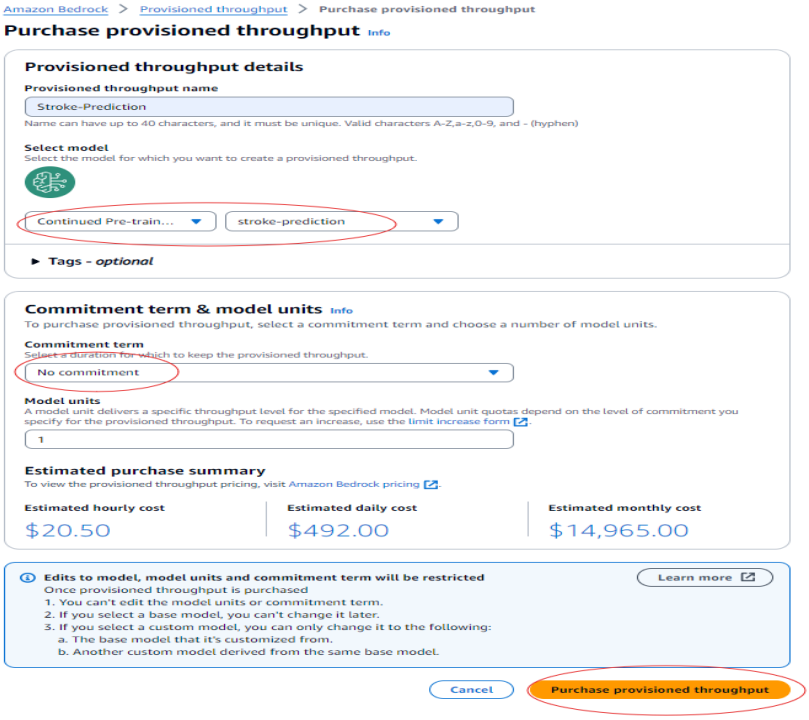


Figure 16. Bedrock Provisioned throughput purchase.

As shown in below fig. 17, we need to evaluate the purchased custom model with sample input prompt to check the prediction as model performance.

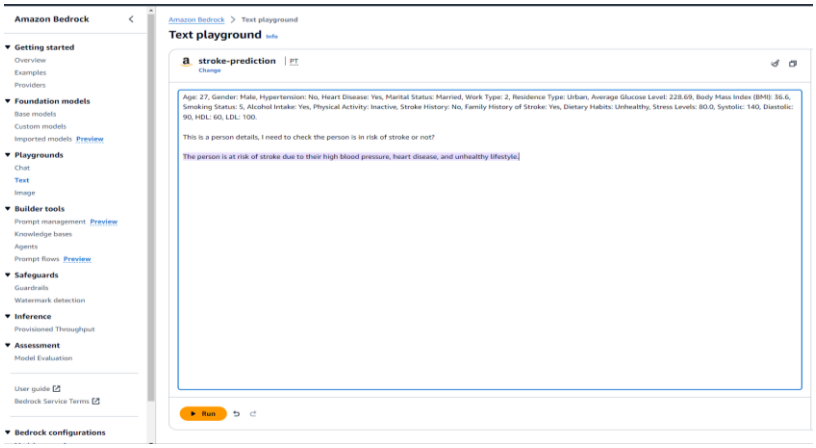

Figure 17. Custom model performance evaluation

**Streamlit:**

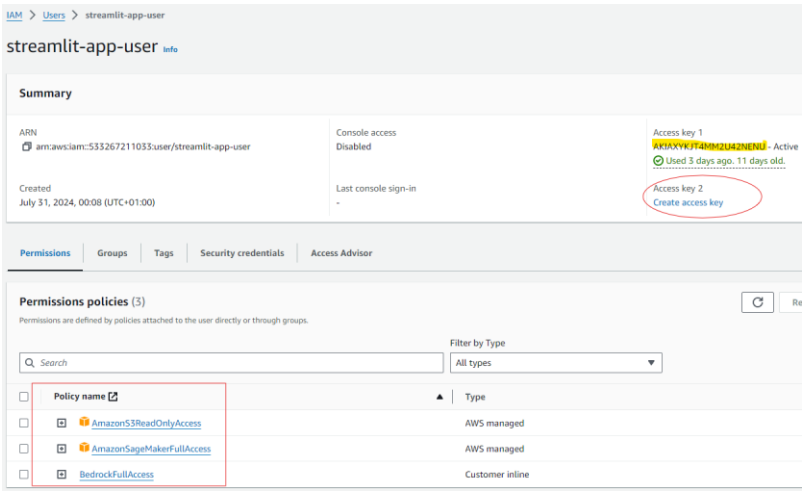

Figure 18. Streamlit app user access permissions

```python
# Set AWS credentials explicitly (not recommended for production)
aws_access_key_id = 'AKIAXYKJT4MM2U42NENU'
aws_secret_access_key = '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'
region_name = 'us-west-2'

# Initialize the SageMaker runtime client with explicit credentials
sagemaker_runtime = boto3.client(
    'sagemaker-runtime',
    region_name=region_name,
    aws_access_key_id=aws_access_key_id,
    aws_secret_access_key=aws_secret_access_key
)

# Initialize the Bedrock client
bedrock_client = boto3.client(
    'bedrock-runtime',
    region_name=region_name,
    aws_access_key_id=aws_access_key_id,
    aws_secret_access_key=aws_secret_access_key
)

# Define the SageMaker endpoint name
sagemaker_endpoint_name = 'xgboost-2024-07-31-21-19-13-748'

# Define the Bedrock model ID
bedrock_model_id = 'arn:aws:bedrock:us-west-2:533267211033:provisioned-model/9at7sbwbnj7t'
```

Figure 19. Integration of SageMaker & Bedrock using endpoints

As fig. 18 ensures to create the IAM Users access to invoke the endpoints seamlessly. Where setting AWS credentials explicitly to get the access key. Make sure to keep the access key is in active and configured properly. Steps to customize the UI, including setting up input forms and displaying the comparison metrics (inference times, risk predictions, etc.).

```
PS C:\Users\rajar\OneDrive\Desktop\personal\stroke-prediction> python app.py
>>
2024-08-11 01:29:23.866
  Warning: to view this Streamlit app on a browser, run it with the following
  command:

    streamlit run app.py [ARGUMENTS]
2024-08-11 01:29:23.871 Session state does not function when running a script without `streamlit run`
PS C:\Users\rajar\OneDrive\Desktop\personal\stroke-prediction> streamlit run app.py

  You can now view your Streamlit app in your browser.

  Local URL: http://localhost:8502
  Network URL: http://192.168.1.34:8502
```

Figure 20. To run Streamlit application

Using the above commands, we can test the Streamlit web UI to evaluate the stroke prediction (Text generation). Here we can to the comparison for prediction, f1 score, inference time, etc. It can be tested locally.

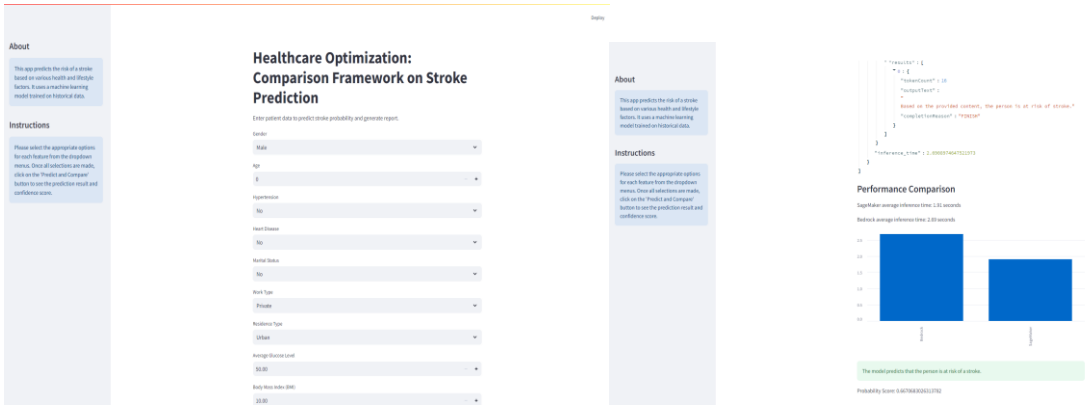


Figure 21. Streamlit application UI for comparison

Here we can get the result like the fig. 21 with the evaluation of risk prediction score, inference time between SageMaker & Bedrock. This how the research outcome we can see.

**Visualizing Results from Stroke prediction model:**

```
# Evaluate the model
from sklearn.metrics import accuracy_score, confusion_matrix, Confusion
import matplotlib.pyplot as plt

accuracy = accuracy_score(y_test, y_pred)
print(f'Accuracy: {accuracy * 100:.2f}%')
```

```
Accuracy: 50.17%
```

```
# Confusion Matrix
cm = confusion_matrix(y_test, y_pred)
disp = ConfusionMatrixDisplay(confusion_matrix=cm)
disp.plot(cmap=plt.cm.Blues)
plt.show()
```

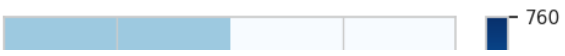


Figure 21. Confusion matrix

```
# Generate predicted probabilities for the positive class
y_pred_proba = []
for index, row in X_test.iterrows():
    result = xgb_predictor.predict(row.values)
    y_pred_proba.append(float(result))  # Directly append the float result

# Convert the list to a numpy array
y_pred_proba = np.array(y_pred_proba)
```

```
import matplotlib.pyplot as plt
from sklearn.metrics import roc_curve, auc, roc_auc_score

# Assuming `y_test` is the true labels and `y_pred_proba` are the predicted probabilities

# Compute ROC curve and AUC
fpr, tpr, thresholds = roc_curve(y_test, y_pred_proba)
roc_auc = auc(fpr, tpr)

# Plotting the ROC curve
plt.figure(figsize=(10, 7))
plt.plot(fpr, tpr, color='darkorange', lw=2, label='ROC curve (area = %0.2f)' % roc_auc)
plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC)')
plt.legend(loc="lower right")
plt.show()

# Print AUC score
print(f'AUC: {roc_auc_score(y_test, y_pred_proba):.2f}')
```

Figure 22. Plotting ROC Curve

Fig. 21, 22 shows the steps followed for plotting the confusion matrix and ROC curve to visualize the accuracy of model. The model implemented in this project is evaluated by the accuracy, precision, recall and F1 score. The confusion matrix was utilized to evaluate values.

# References


Beragu, Suraj (2022) Effective use of Cloud Computing and Machine Learning Technologies for Smart Healthcare Applications. Master's thesis, Dublin, National College of Ireland.

Build generative AI applications on Amazon Bedrock — the secure, compliant, and responsible foundation | Amazon Web Services. (2024, June 29). Amazon Web Services. Available at: https://aws.amazon.com/blogs/machine-learning/build-generative-ai-applications-on-amazon-bedrock-the-secure-compliant-and-responsible-foundation/.

Mohajeri, M.A. (2024). Leveraging large language model for enhanced business analytics on AWS.