

Analysis of cloud environment for implementing machine leaning comparative to local server

MSc Research Project Cloud Computing

Devaraj Devegowda Student ID: 20125429

School of Computing National College of Ireland

Supervisor:

Sean Heeney

National College of Ireland

MSc Project Submission Sheet



School of Computing

Student Devaraj Devegowda Name:			
Student ID	20125429		
Student ID:	Msc cloud computing		2022
Programme:	Msc Research project	Year:	
Module:			
Lecturer	Sean Heeney		
Submission Due Date:	28/04/2022		
Project Title:	Analysis of cloud environment for building m comparative to local server	achine l	eaning services
Word Count:	888 Page Count: 19		

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.

<u>ALL</u> 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.

	Devaraj Devegowda
Signature:	20/04/2022
	28/04/2022
Date:	

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

Devaraj Devegowda 20125429

1 Introduction

This research consists of two development platforms for developing the machine learning. Local machine development and AWS cloud. In local machine learning models are built using Jupyter notebook. And web application is developed in online IDE Cloud 9. SageMaker is used for same ML models.

1.1 Before you begin

Before starting the project make installed below developer tools.

- Git bash: how to install is available athttps://www.educative.io/edpresso/how-to-install-git-bash-in-windows
- Pyhton SDK : https://realpython.com/installing-python/
- Anaconda for Jupyter Notebook: https://www.datacamp.com/community/tutorials/installing-anaconda-windows
- JMeter: https://www.simplilearn.com/tutorials/jmeter-tutorial/jmeter-installation

2 Machine learning implementation

Open the Anaconda navigator click on the jupyter notebook (Annaconda Jupyter, 2022)



After that jupyter notebook opens in default location

💭 Jupyter	Quit Logout
Files Running Clusters	
Select items to perform actions on them.	Upload New - 2
0 • 1	Name Last Modified File size
	a year ago
Applications (Parallels)	10 months ago
	a year ago
Desktop	37 minutes ago
Documents	3 hours ago
Downloads	36 minutes ago
C eclipse	9 months ago
Cir eclipse-workspace	9 months ago
C env	a year ago

2.1 Crop recommendation

```
In [1]: # Importing libraries
         from __future__ import print_function
         import pandas as pd
         import numpy as np
         import matplotlib.pyplot as plt
         import seaborn as sns
         from sklearn.metrics import classification_report
         from sklearn import metrics
         from sklearn import tree
         import warnings
         #For interactivity
         from ipywidgets import interact
         warnings.filterwarnings('ignore')
In [2]: PATH = 'Crop_recommendation.csv'
         df = pd.read_csv(PATH)
In [3]: df.head()
Out[3]:
            N P K temperature humidity
                                             ph
                                                    rainfall label
         0 90 42 43
                       20.879744 82.002744 6.502985 202.935536
                                                           rice
          1 85 58 41
                       21.770462 80.319644 7.038096 226.655537
                                                           rice
          2 60 55 44
                       23.004459 82.320763 7.840207 263.964248
                                                           rice
          3 74 35 40
                       26.491096 80.158363 6.980401 242.864034
                                                           rice
                       20.130175 81.604873 7.628473 262.717340 rice
          4 78 42 42
```

At first we are importing all the necessary libraries and data set. Data set is stored in df variable.

In [10]:	<pre>#Checking the Statistics for all the crops print("Average Ratio of nitrogen in the soil : {0: .2f}".format(df['N'].mean())) print("Average Ratio of Phosphorous in the soil : {0: .2f}".format(df['P'].mean())) print("Average Ratio of Potassium in the soil : {0: .2f}".format(df['K'].mean())) print("Average temperature in Celsius : {0: .2f}".format(df['temperature'].mean())) print("Average Relative Humidity in % is : {0: .2f}".format(df['temperature'].mean())) print("Average pH value of the soil : {0: .2f}".format(df['humidity'].mean())) print("Average Rain fall in mm : {0: .2f}".format(df['rainfall'].mean()))</pre>
	Average Ratio of nitrogen in the soil : 50.55 Average Ratio of Phosphorous in the soil : 53.36 Average Ratio of Potassium in the soil : 48.15 Average temperature in Celsius : 25.62 Average Relative Humidity in % is : 71.48 Average pH value of the soil : 6.47 Average Rain fall in mm : 103.46

Above shows that nitrogen, Phosphorous and potassium should be around 50%

Temperature should be around 25°C and Humidity around 70%

Rain fall should be around 100mm and PH should be arount 7

In	[11]:	df['la	abel'].v	/alue_co	ounts()	
0ut	[11]:	rice		100		
		maize		100		
		jute		100		
		cotto	n	100		
		cocon	ut	100		
		papaya	а	100		
		orange	e	100		
		apple		100		
		muskme	elon	100		
		water	nelon	100		
		grapes	5	100		
		mango		100		
		banana	а	100		
		pomeg	ranate	100		
		lenti	l	100		
		black	gram	100		
		mungb	ean	100		
		mothbe	eans	100		
		pigeo	npeas	100		
		kidne	ybeans	100		
		chick	pea	100		
		coffee	e	100		
		Name:	label,	dtype:	int64	

Above functions shows the number crops in data set.

Distribution for Agricultural Conditions



Above Graph shows us many hidden patterns like many crops need Phosphorous and Potassium at very high level.

Before the doing the modelling data is divided into two one is training and other for testing. In our dataset we are using 20 percent for testing and 80 percent for training.

We are using three different Modelling Classification algorithms

- 1. Decision Tree
- 2. Logistic Regression
- 3. Random Forest



From the above output we can see that random forest gives more accuracy so we will use random forest for predicting the crop. Sample output is given below.

```
In [33]: accuracy_models = dict(zip(model, acc))
for k, v in accuracy_models.items():
    print (k, '-->', v)
Decision Tree --> 0.8590909090909090
Logistic Regression --> 0.96212121212122
RF --> 0.98939393939394
```

Making a prediction

```
In [34]: data = np.array([[50,18, 67, 23.603016, 60.3, 6.7, 140.91]])
prediction = RF.predict(data)
print(prediction)
['pomegranate']
In [35]: data = np.array([[83, 45, 60, 28, 70.3, 7.0, 150.9]])
prediction = RF.predict(data)
print(prediction)
['jute']
```

2.2 Crop yield prediction

At first, we imported important libraries and dataset. In this modelling we are using four different datasets which are

- 1. Yield.csv
- 2. Temperature.csv
- 3. Rainfall.csv
- 4. Pesticides.csv

```
In [1]: import numpy as np
          import pandas as pd
In [2]: df_yield = pd.read_csv('yield.csv')
         df_yield.shape
Out[2]: (56717, 12)
In [3]: df_yield.head()
Out[3]:
             Domain Code Domain Area Code
                                                 Area Element Code Element Item Code
                                                                                      Item Year Code
                                                                                                            Unit Value
                                                                                                     Year
          0
                     QC
                           Crops
                                         2 Afghanistan
                                                              5419
                                                                      Yield
                                                                                  56 Maize
                                                                                                 1961
                                                                                                     1961 hg/ha
                                                                                                                 14000
                      QC
                                                                                  56 Maize
          1
                           Crops
                                         2 Afghanistan
                                                              5419
                                                                      Yield
                                                                                                 1962
                                                                                                     1962 hg/ha
                                                                                                                 14000
                                                                                                      1963 hg/ha
          2
                      QC
                           Crops
                                         2 Afghanistan
                                                              5419
                                                                      Yield
                                                                                  56 Maize
                                                                                                 1963
                                                                                                                 14260
          3
                     QC
                           Crops
                                         2 Afghanistan
                                                              5419
                                                                      Yield
                                                                                  56 Maize
                                                                                                 1964 1964 hg/ha 14257
          4
                     QC
                                                                                  56 Maize
                           Crops
                                         2 Afghanistan
                                                              5419
                                                                      Yield
                                                                                                1965 1965 hg/ha 14400
```

In [6]:	<pre>: # drop unwanted columns. df_yield = df_yield.drop(['Year Code','Element Code','Element','Year Code','Area Code','Domain Code','Domain','U df_yield.head()</pre>					
Out[6]:						
	_	Area	Item	Year	hg/ha_yield	
	0	Afghanistan	Maize	1961	14000	
	1	Afghanistan	Maize	1962	14000	
	2	Afghanistan	Maize	1963	14260	
	3	Afghanistan	Maize	1964	14257	
	4	Afghanistan	Maize	1965	14400	

Removing all other columns which are not needed.

Rainfall

.

In	[9]:	df_rain = pd.read_csv('rainfall.csv')
		df_rain.head()

Out[9]:

	Area	Year	average_rain_fall_mm_per_year
0	Afghanistan	1985	327
1	Afghanistan	1986	327
2	Afghanistan	1987	327
3	Afghanistan	1989	327
4	Afghanistan	1990	327

Now we need to merge the rainfall data with yield data.

In [15]: # merge yield dataframe with rain dataframe by year and area columns yield_df = pd.merge(df_yield, df_rain, on=['Year', 'Area'])

Now, we view the final shape of the dataframe and info of values:

In [16]:	yield_df.shape
Out[16]:	(25385, 5)
In [17]:	yield_df.head()
Out[17]:	

	Area	Item	Year	hg/ha_yield	average_rain_fall_mm_per_year
0	Afghanistan	Maize	1985	16652	327.0
1	Afghanistan	Potatoes	1985	140909	327.0
2	Afghanistan	Rice, paddy	1985	22482	327.0
3	Afghanistan	Wheat	1985	12277	327.0
4	Afghanistan	Maize	1986	16875	327.0

Final dataset is shown below.

Data Exploration

yield_df is the final obtained dataframe;

```
In [33]: yield_df.groupby('Item').count()
```

Itom

Out[33]:

Area Year hg/ha_yield average_rain_fall_mm_per_year pesticides_tonnes avg_temp

nem						
Cassava	2045	2045	2045	2045	2045	2045
Maize	4121	4121	4121	4121	4121	4121
Plantains and others	556	556	556	556	556	556
Potatoes	4276	4276	4276	4276	4276	4276
Rice, paddy	3388	3388	3388	3388	3388	3388
Sorghum	3039	3039	3039	3039	3039	3039
Soybeans	3223	3223	3223	3223	3223	3223
Sweet potatoes	2890	2890	2890	2890	2890	2890
Wheat	3857	3857	3857	3857	3857	3857
Yams	847	847	847	847	847	847

Data Preprocessing

```
In [41]: from sklearn.preprocessing import OneHotEncoder
```

```
In [42]: yield_df_onehot = pd.get_dummies(yield_df, columns=['Area',"Item"], prefix = ['Country',"Item"])
features=yield_df_onehot.loc[:, yield_df_onehot.columns != 'hg/ha_yield']
label=yield_df['hg/ha_yield']
features.head()
```

Out[42]:

	Year	average_rain_fall_mm_per_year	pesticides_tonnes	avg_temp	Country_Albania	Country_Algeria	Country_Angola	Country_Argentina	Country_Armenia
0	1990	1485.0	121.0	16.37	1	0	0	0	0
1	1990	1485.0	121.0	16.37	1	0	0	0	0
2	1990	1485.0	121.0	16.37	1	0	0	0	0
3	1990	1485.0	121.0	16.37	1	0	0	0	0
4	1990	1485.0	121.0	16.37	1	0	0	0	0

5 rows \times 115 columns

Scaling dataset

```
In [46]: from sklearn.preprocessing import MinMaxScaler
scaler=MinMaxScaler()
features=scaler.fit_transform(features)
```

After dropping year column in addition to scaling all values in features, the resulting array will look something like this :

Training data

The training and test datasets will be separated from the original dataset. Because training the model usually necessitates as many data points as feasible, the data is usually split inequitably. For train/test, the most typical splits are 70/30 or 80/20.

Below code shows how to split data set

from sklearn.model_selection import train_test_split

train_data, test_data, train_labels, test_labels = train_test_split(features, label, test_size=0.3, random_state=42)

2.2.1 Model Comparison & Selection

```
In [51]: from sklearn.metrics import r2_score
          def compare_models(model):
              model_name = model.__class__.__name
              fit=model.fit(train_data,train_labels)
              y_pred=fit.predict(test_data)
              r2=r2_score(test_labels,y_pred)
              return([model_name, r2])
In [52]: from sklearn.ensemble import RandomForestRegressor
          from sklearn.ensemble import GradientBoostingRegressor
          from sklearn import svm
          from sklearn.tree import DecisionTreeRegressor
          models = [
              GradientBoostingRegressor(n_estimators=200, max_depth=3, random_state=0),
               RandomForestRegressor(n_estimators=200, max_depth=3, random_state=0),
              svm.SVR(),
             DecisionTreeRegressor()
          1
In [53]: model_train=list(map(compare_models,models))
In [54]: print(*model_train, sep = "\n")
          ['GradientBoostingRegressor', 0.8965768919264416]
          ['RandomForestRegressor', 0.6842532317855172]
          ['SVR', -0.19543203867357395]
          ['DecisionTreeRegressor', 0.9603891419978052]
In [61]: clf=DecisionTreeRegressor()
         model=clf.fit(train_data,train_labels)
         test_df["yield_predicted"]= model.predict(test_data)
         test_df["yield_actual"]=pd.DataFrame(test_labels)["hg/ha_yield"].tolist()
         test_group=test_df.groupby("Item")
         test_group.apply(lambda x: r2_score(x.yield_actual,x.yield_predicted))
Out[61]: Item
         Cassava
                                0.925671
                                0.885694
         Maize
         Plantains and others
                                0.813427
         Potatoes
                                0.910657
         Rice, paddy
                                0.892735
         Sorghum
                                0.802142
                                0.818423
         Sovbeans
         Sweet potatoes
                                0.840371
         Wheat
                                0.921280
         Yams
                                0.925841
         dtype: float64
```

Actual values vs Predicted value

```
In [62]: # So let's run the model actual values against the predicted ones
```

```
fig, ax = plt.subplots()
ax.scatter(test_df["yield_actual"], test_df["yield_predicted"],edgecolors=(0, 0, 0))
ax.set_xlabel('Actual')
ax.set_ylabel('Predicted')
ax.set_title("Actual vs Predicted")
plt.show()

Actual vs Predicted

300000
B 30000
B 30000
B 30000
B 300000
B 30000
B 30000
B 30000
B 300000
B 3000000
```

300000 200000 100000 0 0 100000 200000 200000 Actual

3 Web application development and deployment

Web application is built python 3 flask framework. Code for the whole application is available at https://github.com/devaraj-ncirl/Crop_recommendation (GitHub, 2022)

To clone the above project use the following command

git clone "https://github.com/devaraj-ncirl/Crop_recommendation"



Create cloud 9 online IDE with python 3 installation

Figure 1: Cloud9 IDE

Clone web application using git command

•	9 File Edit Find View (3o Run Tools Window Support Preview 🕟 Run		
q		🗏 Welcome x 🗎 Untitled1 🔹 🚥 README.md x bash - "ip-172-31-20-1 x 🗎 requirements.bt x 🔶 fertilizer.py		👌 app.py
	Cop_recommend: Cop_recommends Cop_recommendation Data Data	<pre># Importing essential libraries and modules from flask import Flask, render_template, request, Markup import numpy as np import pandas as pd from utils.fertilizer import fertilizer_dic import requests import config j import config j import pickle </pre>		
	Interplates Interplates Interplates fortilizer py fortilizer py fortilizer py fortilizer proceilie mr README.md	<pre>10 # Loading crop recommendation model 12 13 [crop_recommendation_model_path = 'Crop_recommendation/models/RandomForest.pkl' 14 crop_recommendation_model_ = pickle.load(15 open(crop_recommendation_model_path, 'rb')) 16 17 18 # 20 # Custom functions for calculations</pre>		
	requirements.txt			10:1 Python
	 Huntime.txt env 	bash - "p-172-31-20-157. x Crop_recommendation/app x 🕒		
	README.md	Run (5 Command: Crop_recommendation/app.py	🖡 Runr	ner: Python 3
		[[dte]]		

Figure2: cloud 9 IDE with Code

If there are any changes in code commit the code first then push it.

To commit code: git commit -m "message" To push code: git push -origin master

After the full development of application, it is deployed to Elastic Beanstalk

3.1 Elastic beanstalk

Create Python environment in Elastic beanstalk use t2.mciro instance.

All environments C Actions V						Create a new environment		
Q	Filter results matching the dis	splay values				<	1 > ③	
	Environment name 🔺	Health ⊽	Application name ∇	Date created ⊽	Last modified ⊽	URL		
0	Croprecommyieldpred- env	Ok	crop_recomm_yield_pred	2022-04- 08 20:10:12 UTC+0100	2022-04- 28 00:44:00 UTC+0100	Croprecommyieldpred env.eba-jvtxqipi.eu- west- 1.elasticbeanstalk.con	l- code-pipe b62f23a5	

Figure 3: Python environment in elastic Beanstalk

3.2 Code Pipeline

Create code pipeline CI/CD integration (AWS, 2022)

Create source using GitHub

GitHub (Version 2)		
(i)	New GitHub version 2 (app-based) action To add a GitHub version 2 action in CodePipeline, you create a connection, wh access your repository. Use the options below to choose an existing connectio more	iich us n or ci	es GitHub Apps to reate a new one. Learn
onnectio	1 xisting connection that you have already configured, or create a new one and then return	to this f	ask.
Q arn:a	ws:codestar-connections:eu-west-1:796640696573:connection/48edb3! 🗙	or	Connect to GitHub
-			
epositor:	Ready to connect /our GitHub connection is ready for use. / name pository in your GitHub account.		
epository hoose a re Q deva	Ready to connect Your GitHub connection is ready for use.		
epositor hoose a re Q deva account>/	Ready to connect Your GitHub connection is ready for use. In name pository in your GitHub account. raj-ncirl/Crop_recommendation X prepository-name>		
epository hoose a re Q deva account>/ ranch na hoose a br	Ready to connect Your GitHub connection is ready for use. Pname pository in your GitHub account. raj-ncirl/Crop_recommendation X repository-name> me anch of the repository.		

Figure 4: Code pipeline setup

Connect to crop recommendation repo and select main branch

Later on deploy stage select already created Elastic bean stalk enviornment.

At the end it should look like below diagram

5aa339			
Faa770			
5aa3	39	39	39

Figure 5: code pipeline success stage

Croprecommyieldpred-env Croprecommyieldpred-env.eba-jvtxqipi.eu-west- Application name: crop_recomm_yield_pred	I.elasticbeanstalk.com 🔀 (e-k2pwidqxgp)	<i>C</i> Refresh Actions ▼
Health	Running version	Platform
	code-pipeline-1651103019159- b62f23a537ebcd462ea7d86efba 96cdc4090d683	
	Upload and deploy	
Ok		Python 3.8 running on 64bit
Causes		Amazon Linux 2/3.3.12
		Change

Figure6: Successful deployment in Elastic beanstalk

After that once we click on web application it should display the homepage



Figure 7 : Home page of Web application

3.3 Web application working

Web application consist of two main features crop recommendation and yield prediction

In home page nav bar click on crop recommend button form will be given to fill details

	Home Crop	Recommendation Yield Prediction How It
Find out the	most suitable crop to gro	ow in your farm
	Nitrogen	
	Enter the value (example:50)	
	Phosphorous	
	Enter the value (example:50)	
	Pottasium	
	Enter the value (example:50)	
	ph level	
	Enter the value	
	Temperature	
	Enter the value	
	Humidity	
	Enter the value	
	Rainfall (in mm)	
	Enter the value	



After filling all the details click on submit button it should give predicted output.

You should grow mothbeans in your farm

Home Crop Recommendation

How It Work

Figure 9: predicted output

same way you can do with crop yield prediction.

4 Machine learning using AWS SageMaker

Create SageMaker Studio

Create Note instance for Machine learning model development ml.t2.meduim (AWS, 2022)

Note	ebook instances			C Actions	Create notebook instance
Q	Search notebook instances				< 1 > ©
	Name 🗢	Instance	Creation time	Status 🗸	Actions
~				0	

Figure 10: Notebook instance

Next step open Jupyter Notebook

Write the code Crop recommendation algorithm in Sage maker note book

\odot	File Edit View Run Kernel Git Tabs	Settings Help
	+ 🖿 ± C 🚸⁺	Crop_recommendation_de × E startup_prediction.py × AWS_Train.ipynb ×
_	🖿 /	🖻 + 🛠 🗇 🗂 🕨 🔳 C Markdown 🗸 🧿 git
0	Name Last Modified	
	crop_recommendation 4 days ago	[1]: import sagemaker from sagemaker import get execution role
•	🖿 data 3 days ago	How Sugemarker Amport get_skeedaton_rote
v	• 🗖 AWS_Train.ipynb 38 minutes ago	<pre>[2]: sagemaker_session = sagemaker.Session()</pre>
æ	Crop_recommendati 38 minutes ago	121, # Get a SaneMaker_compatible role used by this Notebook Instance
•	startup_prediction.py 38 minutes ago	role = get_execution_role()
3		
		[4]: role
		<pre>[4]: 'arn:aws:iam::796640696573:role/service-role/AmazonSageMaker-ExecutionRole-20220424T231788'</pre>
U		
		Upload the data for training
≣		
		<pre>[6]: train_input = sagemaker_session.upload_data("data")</pre>
233		[7]: train_input
		[/]: 22://2adamayai_an_wa2r=1_1200400202121ngrg.
Fig	ure 11: Notebook overview	

Load the trained data to S3 bucket

Trian the model using sklearn container

```
[8]: from sagemaker.sklearn.estimator import SKLearn
script_path = 'startup_prediction.py'
sklearn = SKLearn(
    entry_point=script_path,
    instance_type="ml.m4.xlarge",
    framework_version="0.20.0",
    py_version="py3",
    role=role,
    use_spot_instances=True,
    max_run=300,
    max_wait=600)
    sagemaker_session=sagemaker_session)
```

Train SKLearn Estimator on Startup data

```
[9]: sklearn.fit({'train': train_input})
2022-04-25 09:15:48 Starting - Starting the training job...ProfilerReport-1650878148: InProgress
...
2022-04-25 09:16:31 Starting - Preparing the instances for training.....
2022-04-25 09:17:34 Downloading - Downloading input data...
2022-04-25 09:18:12 Training - Downloading the training image......
2022-04-25 09:19:12 Uploading - Uploading generated training model2022-04-25 09:19:03,222 sagemaker-containers IN
ported framework sagemaker_sklearn_container.training
2022-04-25 09:19:03,226 sagemaker-training-toolkit INF0 No GPUs detected (normal if no gpus installed)
2022-04-25 09:19:03,721 sagemaker-training-toolkit INF0 No GPUs detected (normal if no gpus installed)
2022-04-25 09:19:03,771 sagemaker-training-toolkit INF0 No GPUs detected (normal if no gpus installed)
2022-04-25 09:19:03,771 sagemaker-training-toolkit INF0 No GPUs detected (normal if no gpus installed)
2022-04-25 09:19:03,771 sagemaker-training-toolkit INF0 No GPUs detected (normal if no gpus installed)
2022-04-25 09:19:03,771 sagemaker-training-toolkit INF0 No GPUs detected (normal if no gpus installed)
2022-04-25 09:19:03,770 sagemaker-training-toolkit INF0 Invoking user script
Training Env:
{
```

Deploy the model

```
: deployment = sklearn.deploy(initial_instance_count=1, instance_type="ml.m4.xlarge")
------!
: deployment.endpoint_name
: 'sagemaker-scikit-learn-2022-04-25-09-20-00-666'
: result=deployment.predict([[21,21, 23, 23.603016, 60.3, 5.5, 198.91]])
: print(result)
['pigeonpeas']
```

Create lambda function for calling SageMaker endpoint

C	ode source Info		
	File Edit Find View Go	Tools	Window Test Test Deploy
Q	Go to Anything (% P)	٦	lambda_function × Execution results × (+)
Environment	 crop-recomm-mode ** lambda_function.py 	7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23	<pre>ENDPOINT_NAME = "sagemaker-scikit-learn-2022-04-25-09-20-00-666" runtime= boto3.client('runtime.sagemaker') def lambda_handler(event, context): print("Received event: " + json.dumps(event, indent=2)) data = json.loads(json.dumps(event)) payload = data['data'] print(payload) response = runtime.invoke_endpoint(EndpointName=ENDPOINT_NAME,</pre>

Figure 12: lambda function

4.1.1 API gateway

Create API Gateway to call lambda function

Resources	^{ons} ▼ ●/ - POS	T - Met	hod Execution				
▼ / POST	rest f	\rightarrow	Method Request Auth: NONE ARN: am:aws:execute-api:eu-west- 1:796640696573:q516d3htoj/*/POST/	\rightarrow	Integration Request Type: LAMBDA Region: eu-west-1	\rightarrow	Lambda cro
	Client	\leftarrow	Method Response HTTP Status: 200 Models: application/json => Empty	-	Integration Response HTTP status pattern: Output passthrough: Yes	÷	p-recomm-model

Figure 13: API Gateway

Output:

Using the postman call the endpoint check the output

POST v https://q516d3htoj.execute-api.eu-west-1.amazonaws.com/mycroprecom	Send	~
Params Authorization Headers (8) Body Pre-request Script Tests Settings	Coo	kies
● none ● form-data ● x-www-form-urlencoded ● raw ● binary ● GraphQL JSON ∨	Bea	utify
1 {*data* :[[98,21, 23, 23.603016, 60.3, 5.5, 198.91]]}		Т
iody Cookies Headers (7) Test Results	ave Respon	nse 🗸
Pretty Raw Preview Visualize JSON V	G	Q
1 "coffee"		

Figure 14: Output of lambda function

5 Performance testing

Install JMeter in Local machine

<u>rne Euro Search Run Options Tools H</u> eip	
Test Plan HTTP Request Graph Results Continue Start Next Thread Loop Stop Thread Stop Test Stop Test	w

Figure 15: JMeter setup

Create two threads one fore AWS SageMaker endpoint and other is for web application

For each Thread add Http request, summary report and graphs.

One end point is

```
API Gateway enpoint: https://q516d3htoj.execute-api.eu-west-
1.amazonaws.com/mycroprecom
Web application endpoint: http://croprecommyieldpred-env.eba-jvtxqipi.eu-west-
1.elasticbeanstalk.com/crop-predict
```

For each experiment keep changing the no of threads

Sample summary report should look like below diagram

Test Plan Aws sage maker HTTP Request Graph Results Cal machine learning HTTP Request Graph Results Graph Results	Summary Report Name: Summary Report Comments:										
	Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	Recei	Sent KB/sec	Avg. Bytes
	TOTAL	1100	18225	289	90836	16384.04	31.82%	9.9/sec	11.12	1.72	1150.4
		1100		200			511027	515/500			

Figure 16: JMeter setup

References

Annaconda Jupyter (no date) *Anaconda*. Available at: https://www.anaconda.com/ (Accessed: April 28, 2022).

AWS (2022) *AWS Sage Maker*, *AWS*. Available at: https://aws.amazon.com/getting-started/hands-on/build-train-deploy-machine-learning-model-sagemaker/ (Accessed: April 28, 2022).

AWS (no date) *AWS code pipline*, 2022. Available at: https://aws.amazon.com/codepipeline/ (Accessed: April 28, 2022).

GitHub (no date) GiHUb, 2022. Available at: https://github.com/ (Accessed: April 28, 2022).