

PROJECT REPORT

BREAST CANCER RISK PREDICTION USING AWS SAGEMAKER

Name : Prabhanjan Kumar

Project: Breast Cancer Risk Prediction Using AWS Sagemaker

Domain: Machine Learning

CONTENTS:

1.INTRODUCTION

1.1 Overview

2.LITERATURE SURVEY

2.1 Existing problem

2.2 Proposed solution

3.THEORETICAL ANALYSIS

3.1 Block diagram

3.2 Hardware/Software designing

4.EXPERIMENTAL INVESTIGATIONS

5.RESULT

6.ADVANTAGES & DISADVANTAGES

7.APPLICATIONS

8. FUTURE SCOPE

9.CONCLUSION

10.BIBLIOGRAPHY

1.INTRODUCTION

1.1 OVERVIEW

Cancer has been characterized as a heterogeneous disease consisting of many different subtypes. The early diagnosis and prognosis of a cancer type have become a necessity in cancer research, as it can facilitate the subsequent clinical management of patients. The importance of classifying cancer patients into high or low risk groups has led many research teams, from the biomedical and the [bioinformatics](#) field, to study the application of machine learning (ML) methods. Therefore, these techniques have been utilized as an aim to model the progression and treatment of cancerous conditions. In addition, the ability of ML tools to detect key features from complex datasets reveals their importance. A variety of these techniques, including Artificial Neural Networks (ANNs), Bayesian Networks (BNs), Support Vector Machines (SVMs) and Decision Trees (DTs) have been widely applied in cancer research for the development of predictive models, resulting in effective and accurate decision making. Even though it is evident that the use of ML methods can improve our understanding of cancer progression, an appropriate level of validation is needed in order for these methods to be considered in the everyday clinical practice. In this work, we present a review of recent ML approaches employed in the modeling of cancer progression. The predictive models discussed here are based on various supervised ML techniques as well as on different input features and data samples.

2.LITERATURE SURVEY:

2.1EXISTING PROBLEM:

Breast cancer is one of the main causes of cancer death worldwide. Early diagnostics significantly increases the chances of correct treatment and survival, but this process is tedious and often leads to a disagreement between pathologists.

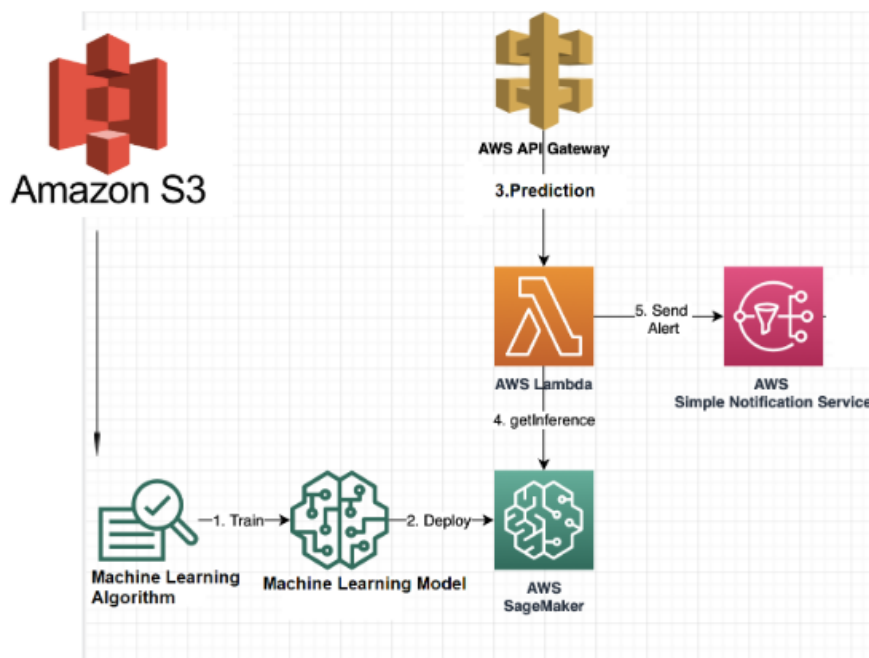
Computer-aided diagnosis systems showed the potential for improving diagnostic accuracy. But early detection and prevention can significantly reduce the chances of death. It is important to detect breast cancer as early as possible. We will be building and deploying the model in AWS SageMaker and use SNS service to generate alerts about the risk.

2.2 PROPOSED SOLUTION:

Develop a model that is capable of detecting the Breast Cancer in early stages. The model must send alerts using the SNS service. The Machine learning model is trained and deployed on Amazon Sage Maker. Create an API Endpoint for the model with the help of API Gateway and AWS Lambda Service

3. THEORETICAL ANALYSIS:

3.1. BLOCK DIAGRAM:



3.2. SOFTWARE DESIGNING:

1. Amazon S3
2. AWS API Gateway
3. AWS Lambda
4. AWS SNS
5. Amazon SageMaker
6. Python 3

4.EXPERIMENTAL INVESTIGATIONS:

Aws Cloud:

Aws Cloud Provides Many Services Such as Sagemaker,lambda and Api Gateway,etc..

Sagemaker:

Amazon SageMaker is a fully managed service that provides every developer and data scientist with the ability to build, train, and deploy machine learning (ML) models quickly. SageMaker removes the heavy lifting from each step of the machine learning process to make it easier to develop high quality models.

Lambda:

With Lambda, you can run code for virtually any type of application or backend service - all with zero administration. Just upload your code and Lambda takes care of everything required to run and scale your code with high availability. You can set up your code to automatically trigger from other AWS services

Api Gateway:

Amazon API Gateway is an AWS service for creating, publishing, maintaining, monitoring, and securing REST, HTTP, and WebSocket APIs at any scale. API developers can create APIs that access AWS or other web services, as well as data stored in the AWS Cloud.API Gateway creates RESTful APIs that:Are

HTTP-based.

Sns:

Amazon Simple Notification Service (SNS) is a fully managed messaging service for both system-to-system and app-to-person (A2P) communication. It enables you to communicate between systems through publish/subscribe (pub/sub) patterns that enable messaging between decoupled microservice applications or to communicate directly to users via SMS, mobile push and email.

5.RESULT:

The screenshot displays a web application titled "Breast Cancer Risk Prediction" running on a local server at 127.0.0.1:5000. The form contains the following input fields:

- Enter Radius Mean
- Enter Perimeter
- Enter Area Mean
- Enter Concave Point Mean
- Enter Radius Worst
- Enter Perimeter Worst
- Enter Area Worst
- Enter Concave Point Worst
- Enter Email Id

The browser's address bar shows the URL 127.0.0.1:5000. The Windows taskbar at the bottom indicates the system time is 14:19 on 07-10-2020.

Student Dashboard x SPS-958-Breast Cancer Risk Pred x Pathways x Sample Form x

127.0.0.1:5000

Apps Neural machine tra... MT Tab-delimited Bilin... Time series forecast... Language Translati... Intuitive Understan... Word Level English... Image captioning w... Text generation wit...

Breast Cancer Risk Prediction

18.87

88.98

800.89

0.17

27.38

189.87

2018.67

0.678

prabha.mathi519@gmail.com

Activate Windows
Go to Settings to activate Windows.

Type here to search

14:20 07-10-2020

Student Dashboard x SPS-958-Breast Cancer Risk Pred x Pathways x Sample Form x

127.0.0.1:5000

Apps Neural machine tra... MT Tab-delimited Bilin... Time series forecast... Language Translati... Intuitive Understan... Word Level English... Image captioning w... Text generation wit...

Breast Cancer Risk Prediction

Result: Benign

Please Consult Doctor Earlier

Enter Radius Mean

Enter Perimeter

Enter Area Mean

Enter Concave Point Mean

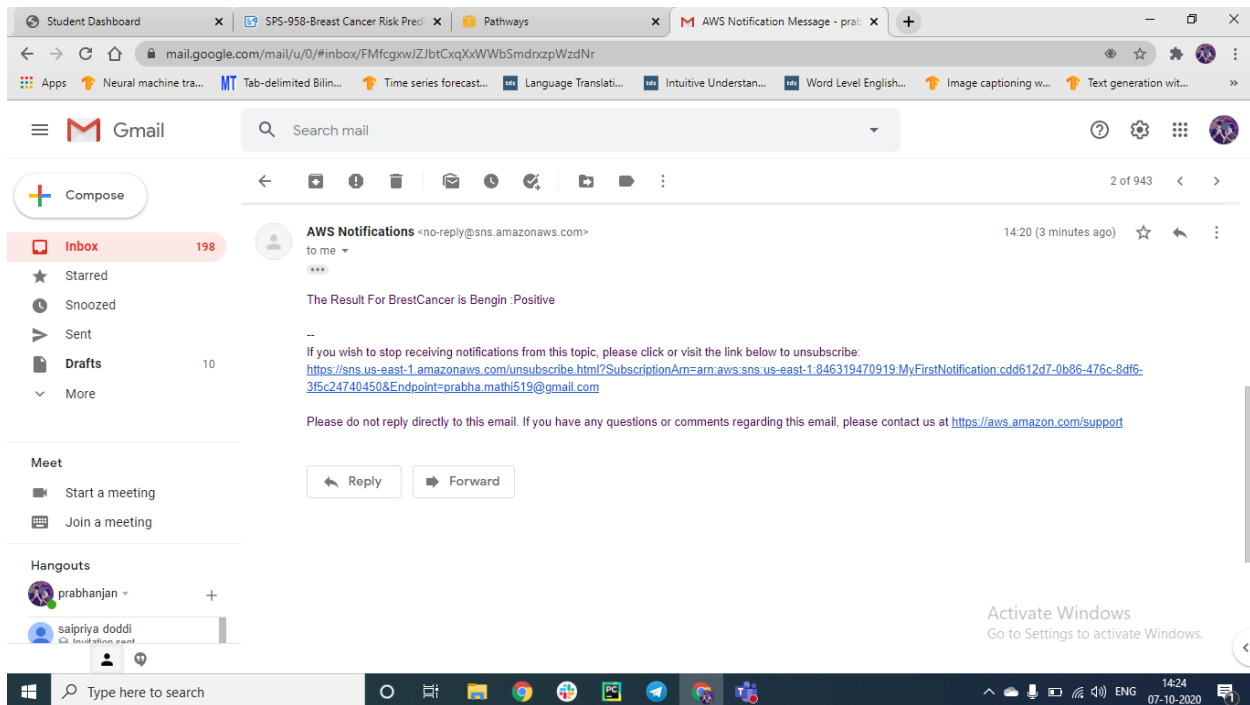
Enter Radius Worst

Enter Perimeter Worst

Activate Windows
Go to Settings to activate Windows.

Type here to search

14:20 07-10-2020



6.ADVANTAGES:

1. It helps to detect the breast cancer in earlier stage
2. It is easy to diagnosis the breast cancer

7.APPLICATIONS:

Health Care Industry

8.FUTURE SCOPE:

The analysis of the results signifies that the integration of multidimensional data along with different classification, feature selection and dimensionality reduction techniques can provide auspicious tools for inference in this domain. Further research in this field should be carried out for the better performance of the classification techniques so that it can predict on more variables. We should intend how to parametrize our classification techniques hence to achieve high accuracy. We should look into many datasets and how further Machine Learning algorithms can be used to characterize Breast Cancer. We want to reduce the error rates with

maximum accuracy.

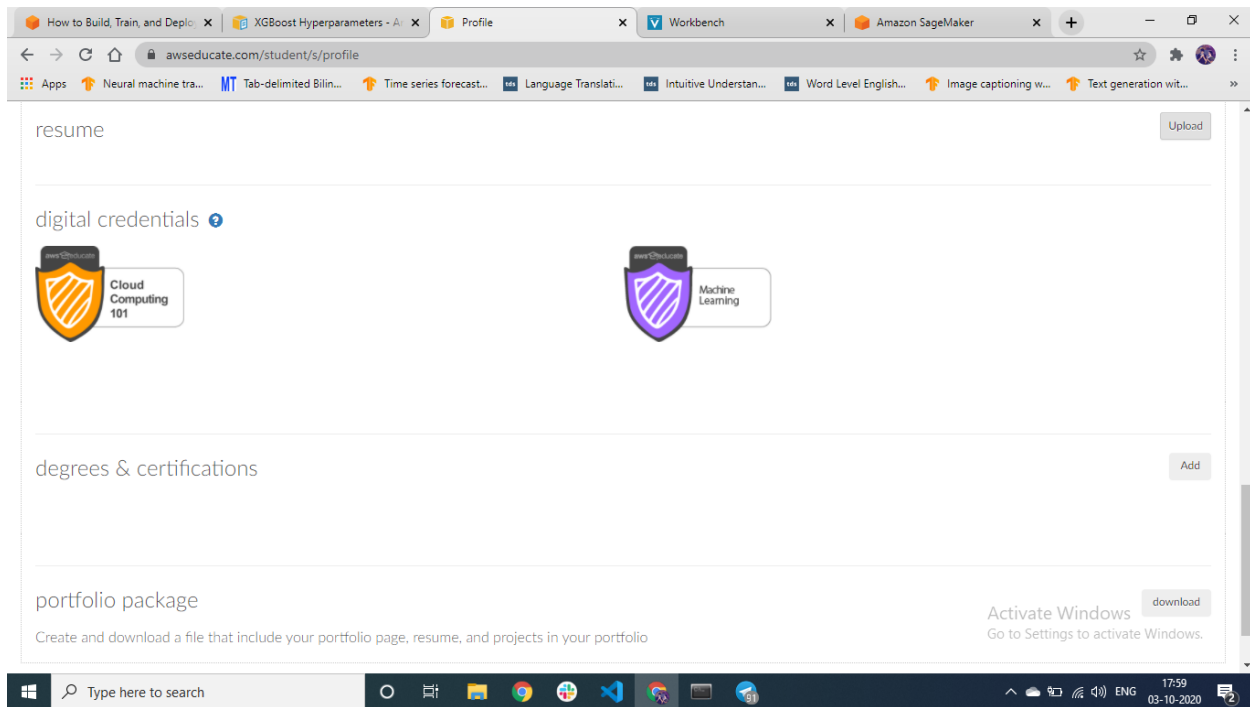
9.CONCLUSION:

There was a striking improvement in the accuracy of classification of women with and without breast cancer achieved with ML algorithms compared to the state-of-the-art model-based approaches. High-accuracy prediction techniques are important in personalized medicine because they facilitate stratification of prevention strategies and individualized clinical management. Predictive models are essential in personalized medicine because they contribute to early identification of high-risk individuals based on known epidemiological and clinical risk factors. Accurate breast cancer risk estimates can inform clinical care and risk management across the breast cancer continuum, e.g., behavioral changes, chemoprevention, personalized screening, and risk-stratified follow-up care. Available risk prediction models have an overall accuracy less than 0.65. ML approaches offer the exciting prospect of achieving improved and more precise risk estimates. This is the first step in developing new risk prediction approaches and further explores diverse risk factors.

10.BIBLIOGRAPHY:

1. Evans DG, Howell A. Breast cancer risk-assessment models. Breast Cancer Res. 2007 Sep 12;9(5):213. pmid:17888188

Carrer PathWay:



Code:

```
1 import numpy as np
2 import pandas as pd
3 import seaborn as sns
4 import matplotlib.pyplot as plt
```

```
1 dataset=pd.read_csv('breastcancer.csv')
2 dataset.head()
```

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave points_mean	...	te
0	842302	M	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.14710	...	
1	842517	M	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0.07017	...	
2	84300903	M	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.12790	...	
3	84348301	M	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0.10520	...	
4	84358402	M	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	0.10430	...	

5 rows × 33 columns



```
1 dataset.shape
```

```
(569, 33)
```

```
1 dataset.info()
```

```

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 569 entries, 0 to 568
Data columns (total 33 columns):
#   Column                                Non-Null Count  Dtype
---  -
0   id                                     569 non-null    int64
1   diagnosis                             569 non-null    object
2   radius_mean                           569 non-null    float64
3   texture_mean                           569 non-null    float64
4   perimeter_mean                         569 non-null    float64
5   area_mean                             569 non-null    float64
6   smoothness_mean                       569 non-null    float64
7   compactness_mean                      569 non-null    float64
8   concavity_mean                        569 non-null    float64
9   concave points_mean                   569 non-null    float64
10  symmetry_mean                         569 non-null    float64
11  fractal_dimension_mean                569 non-null    float64
12  radius_se                             569 non-null    float64
13  texture_se                             569 non-null    float64
14  perimeter_se                           569 non-null    float64
15  area_se                               569 non-null    float64
16  smoothness_se                         569 non-null    float64
17  compactness_se                        569 non-null    float64
18  concavity_se                          569 non-null    float64
19  concave points_se                     569 non-null    float64
20  symmetry_se                           569 non-null    float64
21  fractal_dimension_se                  569 non-null    float64
22  radius_worst                          569 non-null    float64
23  texture_worst                         569 non-null    float64
24  perimeter_worst                       569 non-null    float64
25  area_worst                            569 non-null    float64
26  smoothness_worst                      569 non-null    float64
27  compactness_worst                     569 non-null    float64
28  concavity_worst                       569 non-null    float64
29  concave points_worst                  569 non-null    float64
30  symmetry_worst                        569 non-null    float64
31  fractal_dimension_worst                569 non-null    float64
32  Unnamed: 32                           0 non-null      float64
dtypes: float64(31), int64(1), object(1)
memory usage: 146.8+ KB

```

```

1 dataset=dataset.drop(['Unnamed: 32','id'],axis=1)
2 dataset['diagnosis'].unique()

```

```
array(['M', 'B'], dtype=object)
```

```
1 dataset.isnull().sum()
```

```

diagnosis            0
radius_mean          0
texture_mean         0
perimeter_mean       0
area_mean            0
smoothness_mean      0
compactness_mean     0
concavity_mean       0
concave points_mean  0
symmetry_mean        0
fractal_dimension_mean 0
radius_se            0
texture_se           0
perimeter_se         0
area_se              0
smoothness_se        0
compactness_se       0
concavity_se         0
concave points_se    0
symmetry_se          0
fractal_dimension_se 0
radius_worst         0
texture_worst        0
perimeter_worst      0
area_worst           0
smoothness_worst     0
compactness_worst    0
concavity_worst      0
concave points_worst 0
symmetry_worst       0
fractal_dimension_worst 0
dtype: int64

```

```

1 from sklearn.preprocessing import LabelEncoder
2 le=LabelEncoder()
3 dataset.iloc[:,0]=le.fit_transform(dataset.iloc[:,0]
  ])
4 dataset.head()

```

	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave points_mean	symmetry_mean
0	1	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.14710	0.2419
1	1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0.07017	0.1812
2	1	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.12790	0.2069
3	1	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0.10520	0.2597
4	1	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	0.10430	0.1809

5 rows x 31 columns

```

1 from scipy import stats

```

```

2 corel_points=[]
3 for i in range(31):
4     if i==0:
5         continue
6     else:
7
8         person_cof,p_value=stats.pearsonr(dataset.iloc[:,0]
9         ,dataset.iloc[:,i])
10
11         print('{0} person_cof= {1} and p value =
12         {2} '.format(keys[i],person_cof,p_value))
13
14         if person_cof>0.7:
15             corel_points.append(keys[i])

```

```

radius_mean person_cof= 0.7300285113754564 and p value = 8.465940572258751e-96
texture_mean person_cof= 0.4151852998452044 and p value = 4.058636047896155e-25
perimeter_mean person_cof= 0.742635529725833 and p value = 8.436251036168899e-101
area_mean person_cof= 0.70898383658539 and p value = 4.734564310304486e-88
smoothness_mean person_cof= 0.3585599650859321 and p value = 1.051850359202694e-18
compactness_mean person_cof= 0.5965336775082533 and p value = 3.9382631058850314e-56
concavity_mean person_cof= 0.6963597071719059 and p value = 9.966555755066088e-84
concave points_mean person_cof= 0.7766138400204354 and p value = 7.101150161054297e-116
symmetry_mean person_cof= 0.33049855426254715 and p value = 5.733384028463847e-16
fractal_dimension_mean person_cof= -0.012837602698432378 and p value = 0.7599368037251972
radius_se person_cof= 0.5671338208247177 and p value = 9.738948656455998e-50
texture_se person_cof= -0.008303332973877427 and p value = 0.8433320287665917
perimeter_se person_cof= 0.5561407034314831 and p value = 1.6519051758489454e-47
area_se person_cof= 0.5482359402780244 and p value = 5.895521392602765e-46
smoothness_se person_cof= -0.06701601057948733 and p value = 0.11029660865782694
compactness_se person_cof= 0.29299924424885837 and p value = 9.975994654069982e-13
concavity_se person_cof= 0.25372976598083036 and p value = 8.260176167965638e-10
concave points_se person_cof= 0.4080423327165046 and p value = 3.0723087688164388e-24
symmetry_se person_cof= -0.006521755870647961 and p value = 0.8766418183854183
fractal_dimension_se person_cof= 0.07797241739025615 and p value = 0.06307355082235527
radius_worst person_cof= 0.7764537785950396 and p value = 8.482291921679676e-116
texture_worst person_cof= 0.4569028213967983 and p value = 1.0780574879487261e-30
perimeter_worst person_cof= 0.7829141371737594 and p value = 5.771397139665565e-119
area_worst person_cof= 0.7338250349210511 and p value = 2.828847704284965e-97
smoothness_worst person_cof= 0.42146486106640263 and p value = 6.575143633980474e-26
compactness_worst person_cof= 0.590998237841792 and p value = 7.06981635253457e-55
concavity_worst person_cof= 0.659610210369233 and p value = 2.4646639567816386e-72
concave points_worst person_cof= 0.79356601714127 and p value = 1.9690997072156805e-124
symmetry_worst person_cof= 0.416294311048619 and p value = 2.9511205771522968e-25
fractal_dimension_worst person_cof= 0.3238721887208239 and p value = 2.3164324499817004e-15

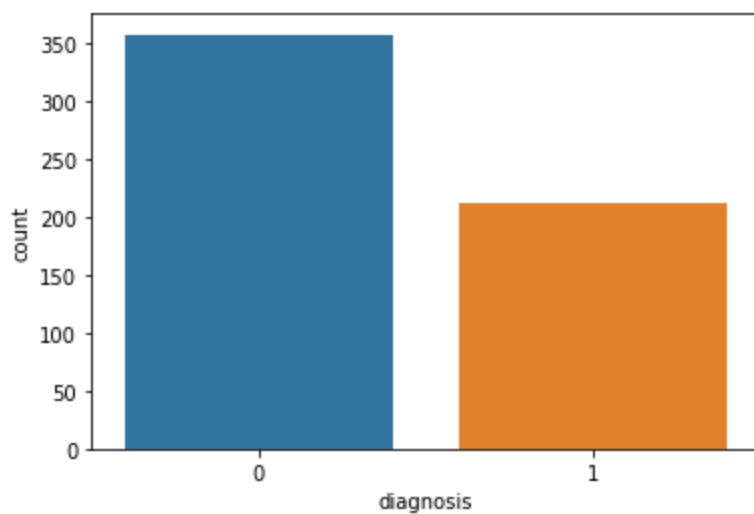
```

```
1 corel_points.append('diagnosis')
2 data=dataset[corel_points]
3 data.head()
```

	radius_mean	perimeter_mean	area_mean	concave points_mean	radius_worst	perimeter_worst	area_worst	concave points_worst	diagnosis
0	17.99	122.80	1001.0	0.14710	25.38	184.60	2019.0	0.2654	1
1	20.57	132.90	1326.0	0.07017	24.99	158.80	1956.0	0.1860	1
2	19.69	130.00	1203.0	0.12790	23.57	152.50	1709.0	0.2430	1
3	11.42	77.58	386.1	0.10520	14.91	98.87	567.7	0.2575	1
4	20.29	135.10	1297.0	0.10430	22.54	152.20	1575.0	0.1625	1

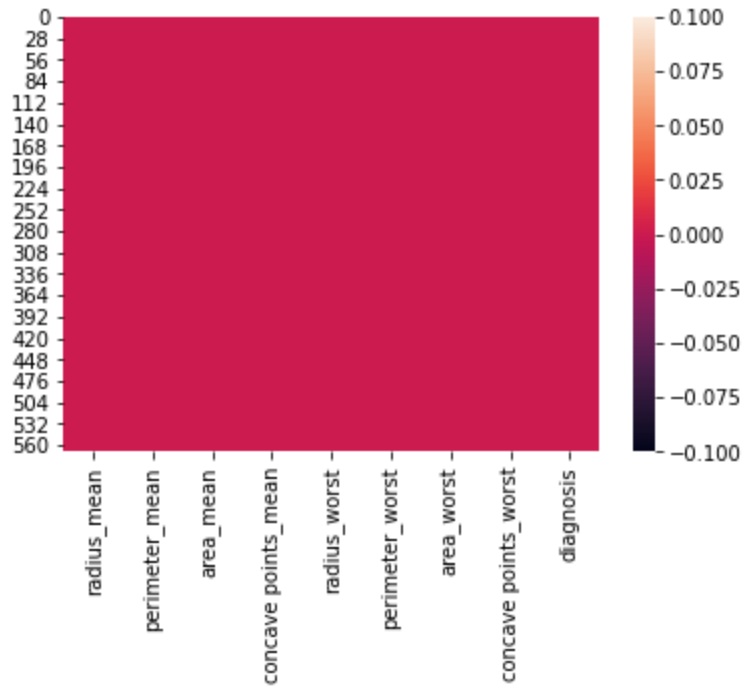
```
1 sns.countplot(x='diagnosis',data=data)
```

<matplotlib.axes._subplots.AxesSubplot at 0x7fef6414a320>



```
1 sns.heatmap(data.isnull())
```

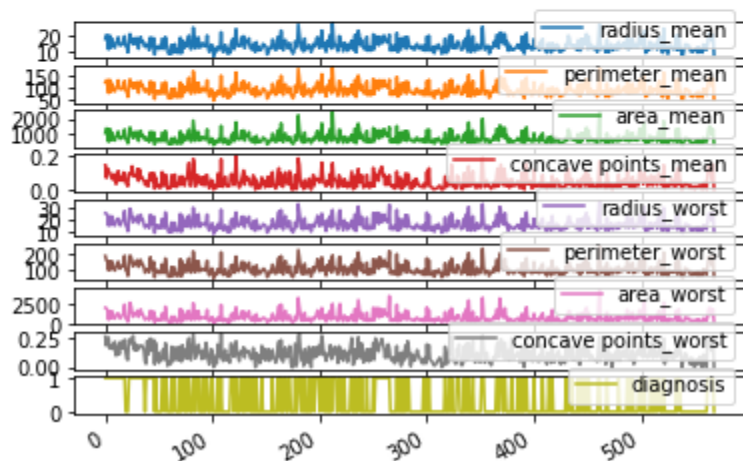
<matplotlib.axes._subplots.AxesSubplot at 0x7fef5f8f8630>



```
1 data.plot(kind='line',subplots=True)
```

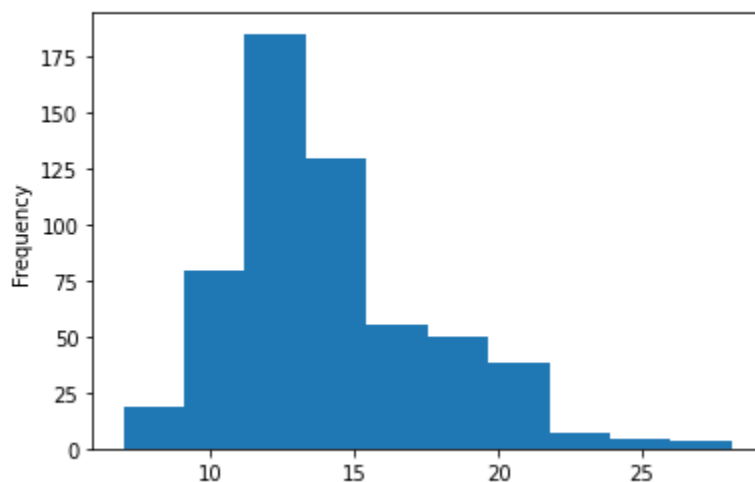


```
array([<matplotlib.axes._subplots.AxesSubplot object at 0x7fef5e8bfa20>,
      <matplotlib.axes._subplots.AxesSubplot object at 0x7fef5e872cc0>,
      <matplotlib.axes._subplots.AxesSubplot object at 0x7fef5e828cc0>,
      <matplotlib.axes._subplots.AxesSubplot object at 0x7fef5e7ddcc0>,
      <matplotlib.axes._subplots.AxesSubplot object at 0x7fef5e793cc0>,
      <matplotlib.axes._subplots.AxesSubplot object at 0x7fef5e7c7cc0>,
      <matplotlib.axes._subplots.AxesSubplot object at 0x7fef5e77ecc0>,
      <matplotlib.axes._subplots.AxesSubplot object at 0x7fef5e734c88>,
      <matplotlib.axes._subplots.AxesSubplot object at 0x7fef5e734cf8>],
      dtype=object)
```



```
1 data['radius_mean'].plot(kind='hist')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fef5e5be2e8>
```

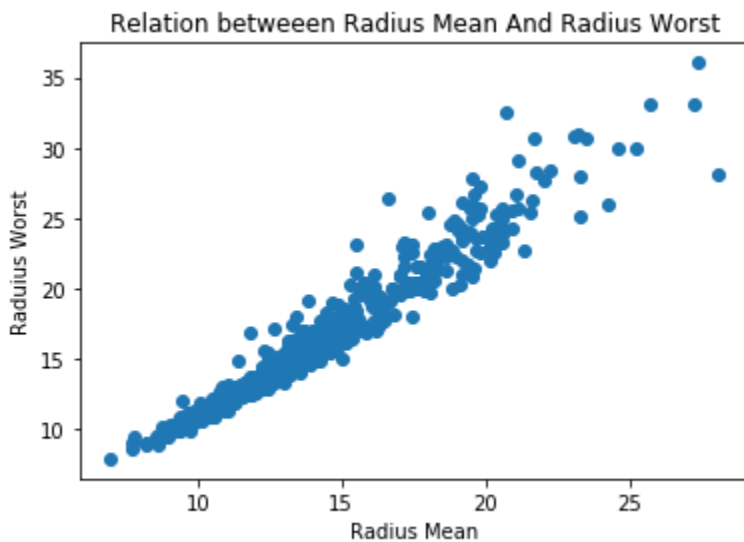


```
1 plt.scatter(dataset['radius_mean'], dataset['radius_
```

```

    worst'])
2 plt.xlabel('Radius Mean')
3 plt.ylabel('Radius Worst')
4 plt.title('Relation between Radius Mean And Radius
    Worst')
5 plt.show()

```



```

1 data_in=data.iloc[:, :-1]
2 data_out=data.iloc[:, -1]

```

```

1 from sklearn.preprocessing import MinMaxScaler
2 from sklearn.model_selection import
    train_test_split

```

```

1 sc=MinMaxScaler(feature_range=(0,1))
2 data_preprocess=sc.fit_transform(data_in)
3 final_keys=data_in.keys()

```

```

4 dicti={}
5 for i in range(len(final_keys)):
6
7     dicti.update({final_keys[i]:data_preprocess[:,i]})
8 dataset=pd.DataFrame(dicti)
9 dataset.head()

```

	radius_mean	perimeter_mean	area_mean	concave points_mean	radius_worst	perimeter_worst	area_worst	concave points_worst
0	0.521037	0.545989	0.363733	0.731113	0.620776	0.668310	0.450698	0.912027
1	0.643144	0.615783	0.501591	0.348757	0.606901	0.539818	0.435214	0.639175
2	0.601496	0.595743	0.449417	0.635686	0.556386	0.508442	0.374508	0.835052
3	0.210090	0.233501	0.102906	0.522863	0.248310	0.241347	0.094008	0.884880
4	0.629893	0.630986	0.489290	0.518390	0.519744	0.506948	0.341575	0.558419

```

1 final_data=pd.concat([data.iloc[:,-1],dataset],axis
2                        =1)
3 train,test=train_test_split(final_data,test_size=0.
4                               2)

```

```

1 import boto3,re,os,json,sagemaker
2 from sagemaker import get_execution_role

```

```

1 role=get_execution_role()
2 my_region=boto3.session.Session().region_name

```

```

1 containers = {'us-west-2':
2               '433757028032.dkr.ecr.us-west-2.amazonaws.com/xgboost:latest',
3               'us-east-1':
4               '811284229777.dkr.ecr.us-east-1.amazonaws.com/xgboost:latest',

```

```
3         'us-east-2':  
    '825641698319.dkr.ecr.us-east-2.amazonaws.com/xgboost:latest',  
4         'eu-west-1':  
    '685385470294.dkr.ecr.eu-west-1.amazonaws.com/xgboost:latest'}
```

```
1 prefix='sagemaker/Brest-Cancer'  
2 bucket_name='buildathonproject1'
```

```
1 final_data.to_csv('train.csv',index=False,header=False)  
2 boto3.Session().resource('s3').Bucket(bucket_name).Object(os.path.join(prefix,'train/train.csv')).upload_file('train.csv')  
3 s3_input_train=sagemaker.s3_input(s3_data='s3://{}/{}/train'.format(bucket_name, prefix),content_type='csv')
```

```
1 sess=sagemaker.Session()  
2 brest_cancer_model=sagemaker.estimator.Estimator(containers[my_region],role,train_instance_count=1,train_instance_type='ml.m5.large',output_path='s3://{}/{}/output'.format(bucket_name,prefix),sagemaker_session=sess)  
3 brest_cancer_model.set_hyperparameters(max_depth=5,eta=0.2,gamma=4,min_child_weight=6,subsample=0.8,silent=0,objective='binary:logistic',num_round=100)
```

```
1 breast_cancer_model.fit({'train':s3_input_train})
```

```
2020-09-28 02:59:12 Starting - Starting the training job...
2020-09-28 02:59:15 Starting - Launching requested ML instances.....
2020-09-28 03:00:46 Starting - Preparing the instances for training...
2020-09-28 03:01:32 Downloading - Downloading input data...
2020-09-28 03:02:04 Training - Downloading the training image..Arguments: train
[2020-09-28:03:02:20:INFO] Running standalone xgboost training.
[2020-09-28:03:02:20:INFO] Path /opt/ml/input/data/validation does not exist!
[2020-09-28:03:02:20:INFO] File size need to be processed in the node: 0.09mb. Available memory size in the node: 255.09mb
[2020-09-28:03:02:20:INFO] Determined delimiter of CSV input is ','
[03:02:20] S3DistributionType set as FullyReplicated
[03:02:20] 569x8 matrix with 4552 entries loaded from /opt/ml/input/data/train?format=csv&label_column=0&delimiter=,
[03:02:20] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 4 extra nodes, 4 pruned nodes, max_depth=2
[0]#011train-error:0.056239
[03:02:20] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned nodes, max_depth=2
[1]#011train-error:0.050967
[03:02:20] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 2 pruned nodes, max_depth=2
[2]#011train-error:0.052724
[03:02:20] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 4 extra nodes, 2 pruned nodes, max_depth=2
[3]#011train-error:0.050967
```

```
1 detector=breast_cancer_model.deploy(initial_instance
_count=1,instance_type='ml.m5.large')
```

Parameter image will be renamed to image_uri in SageMaker Python SDK v2.

-----!

```
1 detector.endpoint
```

```
'xgboost-2020-09-28-02-59-12-505'
```

```
1 from sagemaker.predictor import csv_serializer
2 test_data_array=test.drop('diagnosis',axis=1).value
  s #load the data into an array
3 detector.content_type = 'text/csv' # set the data
  type for an inference
4 detector.serializer = csv_serializer # set the
  serializer type
5 predictions=detector.predict(test_data_array).decod
  e('utf-8') # predict!
6 predictions_array = np.fromstring(predictions[1:],
  sep=',')
7 print(predictions)
```

0.984164893627,0.984164893627,0.984164893627,0.00840793922544,0.00840793922544,0.984164893627,0.164673551917,0.00840793922544,0.698992073536,0.984164893627,0.00840793922544,0.00840793922544,0.00840793922544,0.00840793922544,0.00840793922544,0.984164893627,0.00840793922544,0.0126698166132,0.81916064024,0.984164893627,0.00840793922544,0.00840793922544,0.00840793922544,0.984164893627,0.880655944347,0.00840793922544,0.980865299702,0.00840793922544,0.984164893627,0.0269436519593,0.119824945927,0.00840793922544,0.733735799789,0.00840793922544,0.984164893627,0.0277226809412,0.984164893627,0.0302205681801,0.00840793922544,0.00840793922544,0.0277226809412,0.0715783983469,0.975009799004,0.0317783281207,0.477484881878,0.00840793922544,0.984164893627,0.00840793922544,0.950887620449,0.00840793922544,0.00840793922544,0.00840793922544,0.984164893627,0.0269436519593,0.984164893627,0.0317783281207,0.00840793922544,0.00840793922544,0.0180823151022,0.00840793922544,0.950887620449,0.340621441603,0.975009799004,0.980555832386,0.00840793922544,0.980555832386,0.984164893627,0.00840793922544,0.929939746857,0.00840793922544,0.00840793922544,0.984164893627,0.950887620449,0.582483589649,0.0180823151022,0.00840793922544,0.00840793922544,0.211229816079,0.0201756022871,0.984164893627,0.00840793922544,0.137797668576,0.0126698166132,0.00840793922544,0.0492947809398,0.00840793922544,0.966929972172,0.00840793922544,0.00840793922544,0.984164893627,0.0634867548943,0.175529614091,0.984164893627,0.0513216592371,0.00840793922544,0.942702412605,0.00840793922544,0.00840793922544,0.00840793922544,0.00840793922544,0.303190141916,0.0126698166132,0.984164893627,0.00840793922544,0.154683455825,0.00840793922544,0.984164893627,0.0277226809412,0.00840793922544,0.00840793922544,0.0277226809412,0.0180823151022

LambdaCode:

```
1 import os
2 import io
3 import boto3
4 import json
5 import csv
6 def lambda_handler(event, context):
7     ENDPOINT_NAME =
8     os.environ['environment_variable']
9     runtime= boto3.client('runtime.sagemaker')
10    print(ENDPOINT_NAME)
11    print("Received event: " , json.dumps(event,
12        indent=2))
13    data = json.loads(json.dumps(event))
14    print("Data:",data)
15    payload = data['data']
16    print("Payload:",payload)
17    response =
18    runtime.invoke_endpoint(EndpointName=ENDPOINT_NAME,
19        ContentType='text/csv',
```

```

    Body=payload)
18     result =
    json.loads(response['Body'].read().decode())
19     sns = boto3.client('sns')
20
    sub=sns.subscribe(TopicArn='arn:aws:sns:us-east-1:8
46319470919:MyFirstNotification',Protocol='email',E
ndpoint=data['email'])
21
22     print(result)
23
24     if result>0.5:
25         response =
    sns.publish(TopicArn='arn:aws:sns:us-east-1:8463194
70919:MyFirstNotification',Message='The Result For
BrestCancer is Benign :Positive',)
26         return "Benign"
27     else:
28         response =
    sns.publish(TopicArn='arn:aws:sns:us-east-1:8463194
70919:MyFirstNotification',Message='The Result For
BrestCancer is Malignant :Negitive',)
29         return "Malignant"

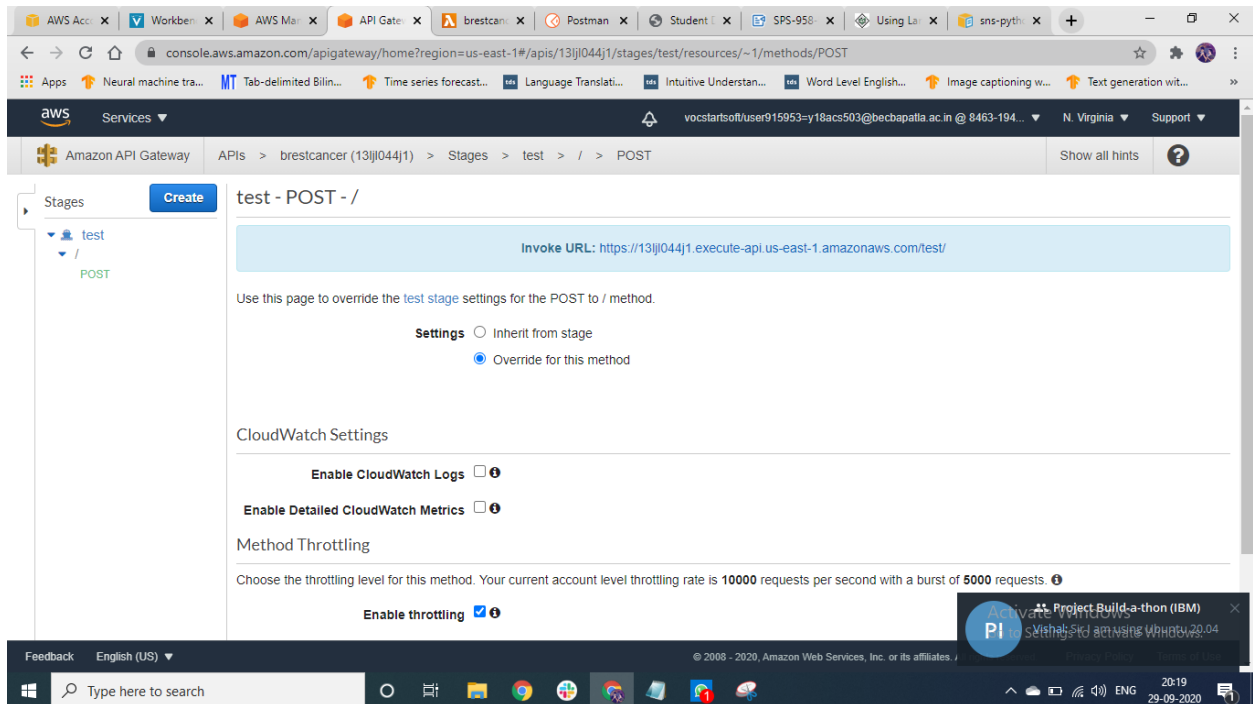
```

```

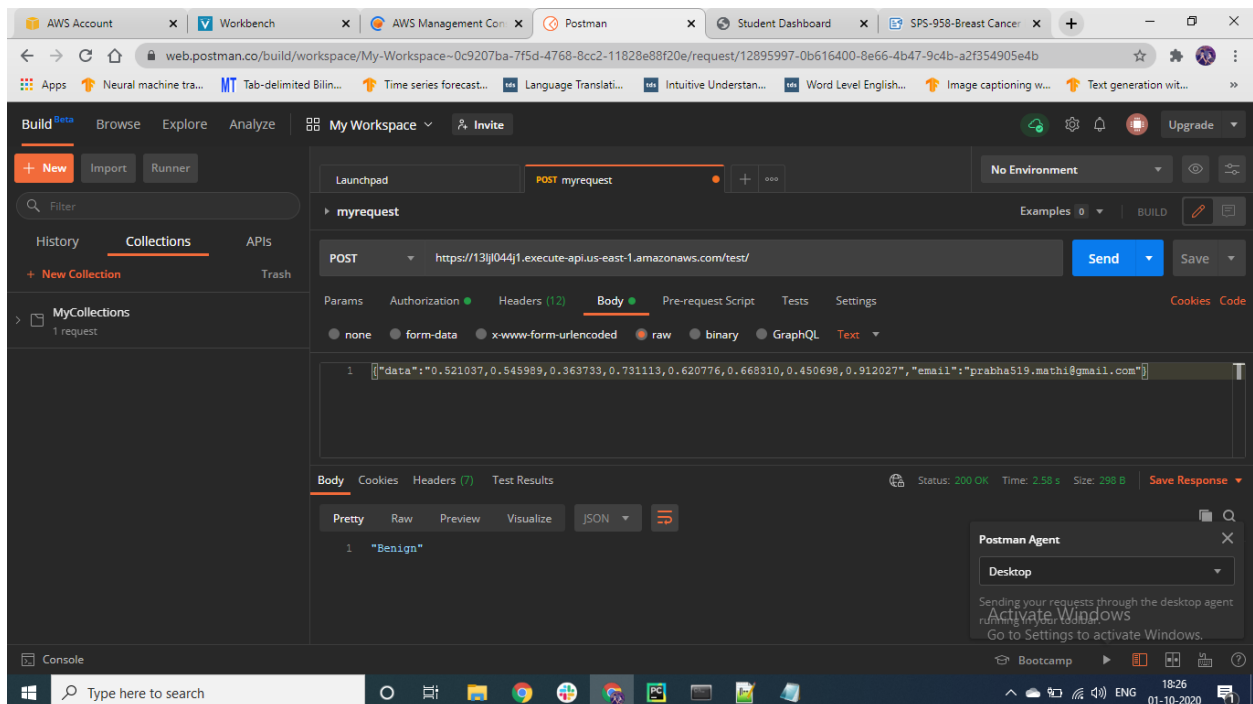
Response:
"Benign"

```

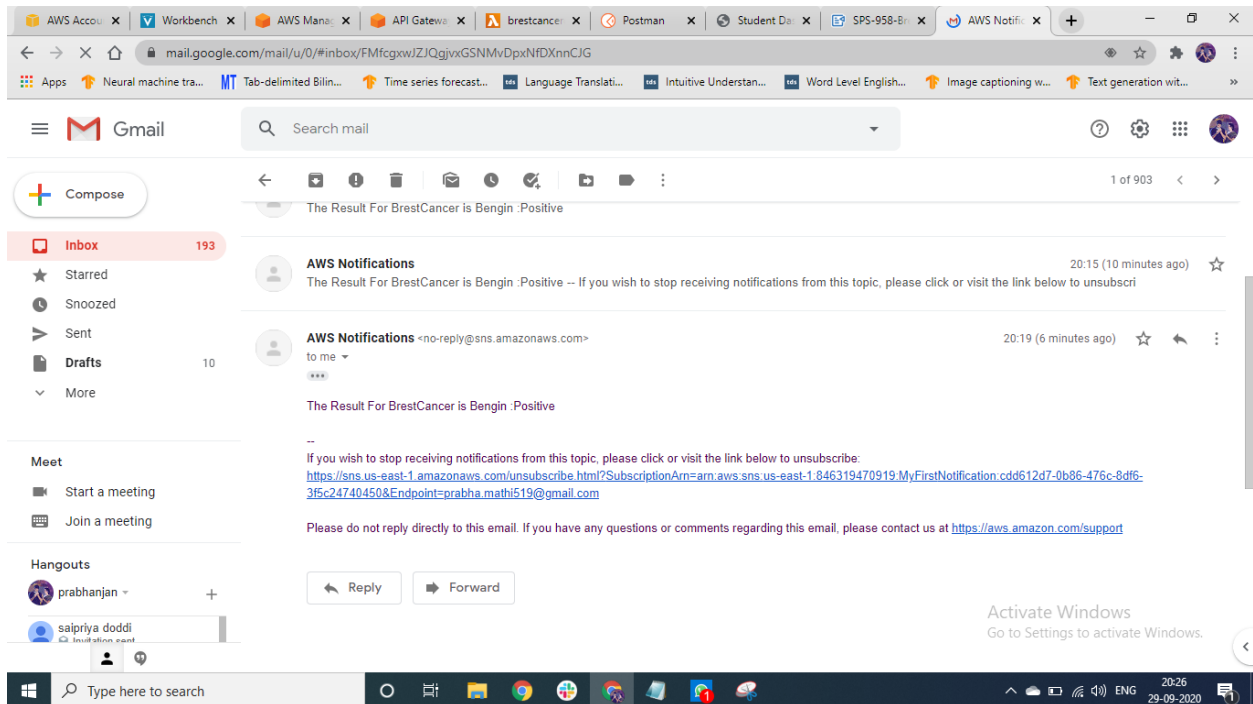
Creation Of Api:



Testing Api:



EmailNotification:



UI: hello.py

```
1 from flask import
   Flask,render_template,request,url_for
2 import requests
3 app=Flask(__name__)
4 @app.route('/',methods=['POST','GET'])
5 def hello():
6     if request.method=='POST':
7         radme=request.form['a']
8         permn=request.form['b']
9         armn=request.form['c']
10        compm=request.form['d']
11        radw=request.form['e']
12        perw=request.form['f']
```

```

13         arw=request.form['g']
14         copw=request.form['h']
15         email=request.form['i']
16         try:
17             radme=float(radme)
18             permn=float(permn)
19             armn=float(armn)
20             compm=float(compm)
21             radw=float(radw)
22             perw=float(perw)
23             arw=float(arw)
24             copw=float(copw)
25             email=str(email)
26         except:
27             return
28         render_template('data.html',err_msg='Enter Valid
Data')
29         url =
        "https://13lj1044j1.execute-api.us-east-1.amazonaws
.com/test/"
30         payload = " {"data": " " + str(radme) +
        ', ' + str(permn) + ', ' + str(armn) + ', ' +
        str(compm) + ', ' + str(radw) + ', ' + str(perw) +
        ', ' + str(arw) + ', ' + str(copw) + "\"" + ', ' +
        "\"email\"" + ":" + "\"" + email + "\"" + "}"
31         print(payload)
32         headers = {
33             'X-Amz-Content-Sha256':
        'beaead3198f7da1e70d03ab969765e0821b24fc913697e929e
726aeaebf0eba3',

```

```

34         'X-Amz-Date': '20200930T095337Z',
35         'Authorization': 'AWS4-HMAC-SHA256
    Credential=ASIA4KDESJFDUSSQKJSG/20200930/us-east-1/
    execute-api/aws4_request,
    SignedHeaders=host;x-amz-content-sha256;x-amz-date,
    Signature=b81935cc533d5efb8db465da9c12f4a3ed76ca800
    89dfc3edebdb39df4fe5f7c',
36         'Content-Type': 'text/plain'
37     }
38
39     response = requests.request("POST", url,
    headers=headers, data=payload)
40     response=response.text.encode('utf8')
41     response=str(response)
42     print(response)
43     result=response[3:-2]
44     print(result)
45     if result=="Benign":
46         return
    render_template('data.html',result=result,sug='Plea
    se Consult Doctor Earlier')
47     else:
48         return
    render_template('data.html',result=result,sug='You\
    'r health is Fine')
49     else:
50         return render_template('data.html')
51
52 if __name__ == '__main__':
53     app.run(debug=True)
54

```

data.html:

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4     <title>Sample Form</title>
5     <style>
6         .brest
7         {
8             width:300px;
9             height:25px;
10            background-color:#ffffbf;
11            border-style: ridge;
12            border-color:gray;
13            border-radius:6px;
14        }
15        body
16        {
17            font-family:sans-serif;
18            background-image: url('{{
19                url_for('static', filename='back.png') }}');
20        }
21        #sub
22        {
23            width:200px;
24            height:25px;
25            background-color:#9f6cff;
26            border-style: ridge;
27            border-color:gray;
28            border-radius:6px;
```

```
28     }
29     </style>
30</head>
31<body style="background-color:#575143">
32     <center>
33         <h1 style="color:red;">Breast Cancer Risk
    Prediction</h1>
34         <div style="background-color:#a8a883;">
35             <br/>
36             {% if result %}
37             <p style="color:blue;font-size:30px;">
    Result: {{result}}</p>
38             <p
    style="color:red;font-size:20px;">{{sug}}</p>
39             {% endif %}
40             {% if err_msg %}
41             <p
    style="color:red;font-size:15px;">{{err_msg}}</p>
42             {% endif %}
43             <br/>
44             <form method="post" action="/">
45                 <input type="text" name="a"
    class="breast" placeholder="    Enter Radius Mean"
    required><br/><br/>
46                 <input type="text" name="b"
    class="breast" placeholder="    Enter Perimeter "
    required><br/><br/>
47                 <input type="text" name="c"
    class="breast" placeholder="    Enter Area Mean"
    required><br/><br/>
48                 <input type="text" name="d"
```

```
class="brest" placeholder="    Enter Concave Point  
Mean" required><br/><br/>  
49      <input type="text" name="e"  
class="brest" placeholder="    Enter Radius Worst"  
required><br/><br/>  
50      <input type="text" name="f"  
class="brest" placeholder="    Enter Perimeter  
Worst" required><br/><br/>  
51      <input type="text" name="g"  
class="brest" placeholder="    Enter Area Worst"  
required><br/><br/>  
52      <input type="text" name="h"  
class="brest" placeholder="    Enter Concave Point  
Worst" required><br/><br/>  
53      <input type="email" name="i"  
class="brest" placeholder="    Enter Email Id"  
required><br/><br/>  
54      <input type="submit" value="Submit"  
id="sub">  
55      <br/><br/><br/>  
56      </form>  
57      </div>  
58      </center>  
59</body>  
60</html>
```