PROJECT REPORT

DIABETICS PREDICTION MODELFOR INDIAN WOMEN

Name:Prabhanjan Kumar

Project:Diabetics Prediction Model For Indian Women

Domain: Machine Learning

CONTENTS:

1.INTRODUCTION

1.1 OVERVIEW

Diabetes is a chronic disease with the potential to cause a worldwide health care crisis. However, early prediction of diabetes is quite challenging task for medical practitioners due to complex interdependence on various factors as diabetes affects human organs such as kidney, eye, heart, nerves, foot etc. Data science methods have the potential to benefit other scientific fields by shedding new light on common questions. One such task is to help make predictions on medical data.

This project also aims to propose an effective technique for earlier detection of the diabetes disease.

2.LITERATURE SURVEY:

2.1EXISTING PROBLEM:

In this, we need to diagnosticallypredict whether or not a patient has diabetes, based on certain diagnostic measurements included in the dataset. Several constraints were placed on the selection of these instances from a larger database. In particular, all patients here are females at least 21 years old of Pima Indian heritage.

The datasets consist of several medical predictor variables and one target variable, Diabetes. Predictor variables include the number of pregnancies the patient has had, their BMI, insulin level, age, and so on.

2.2PROPOSED SOLUTION:

Develop an end-to-end web application that predicts the probability of females having diabetes. The application must be built with Python-Flask or Django framework with the machine learning model trained & deployed on AWS Sagemaker. Create an API Endpoint for the model with the help of API Gateway and AWS Lambda Service.

3.THEORITICAL ANALYSIS:

3.1. BLOCK DIAGRAM:

3.2. SOFTWARE DESIGNING:

1. Amazon S3
2. AWS API Gateway
3. AWS Lambda
4. Amazon SageMaker

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.

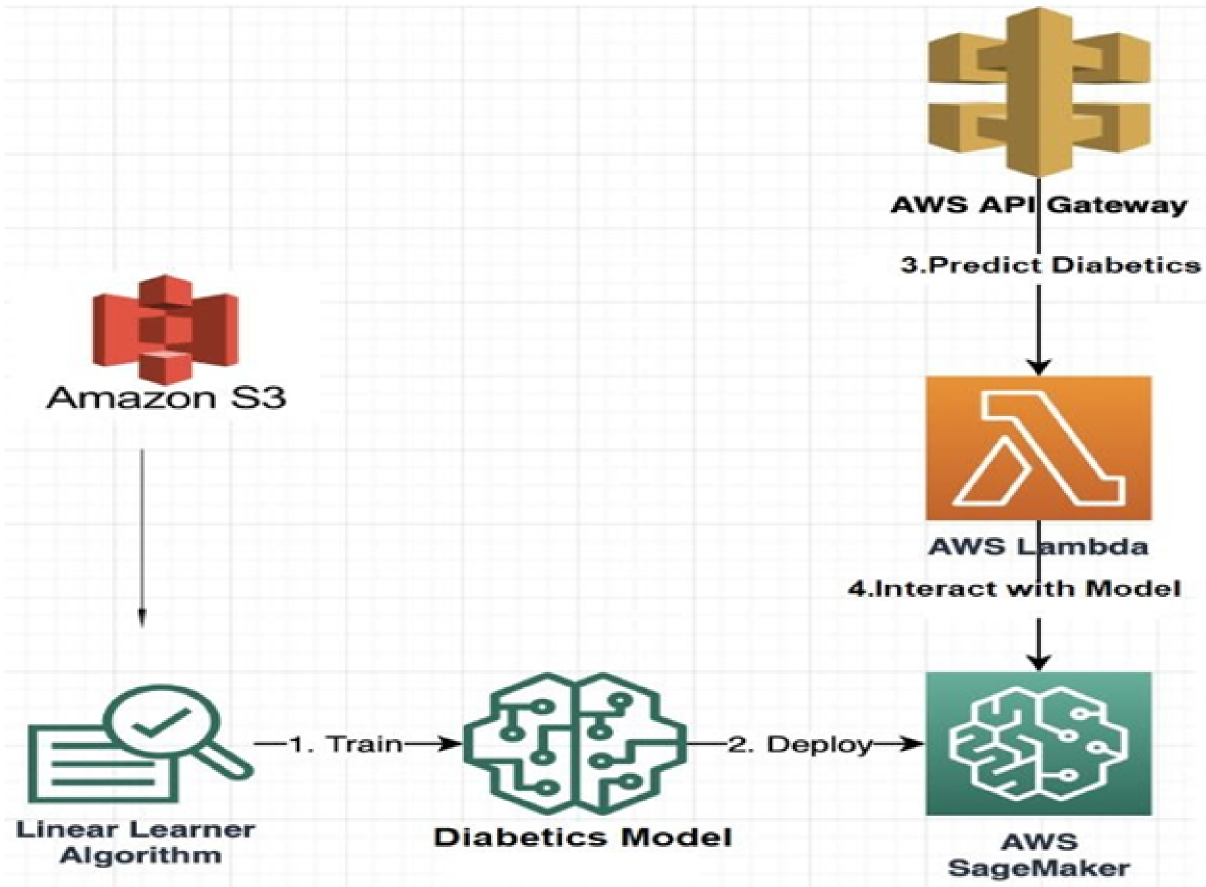
Lambda:

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

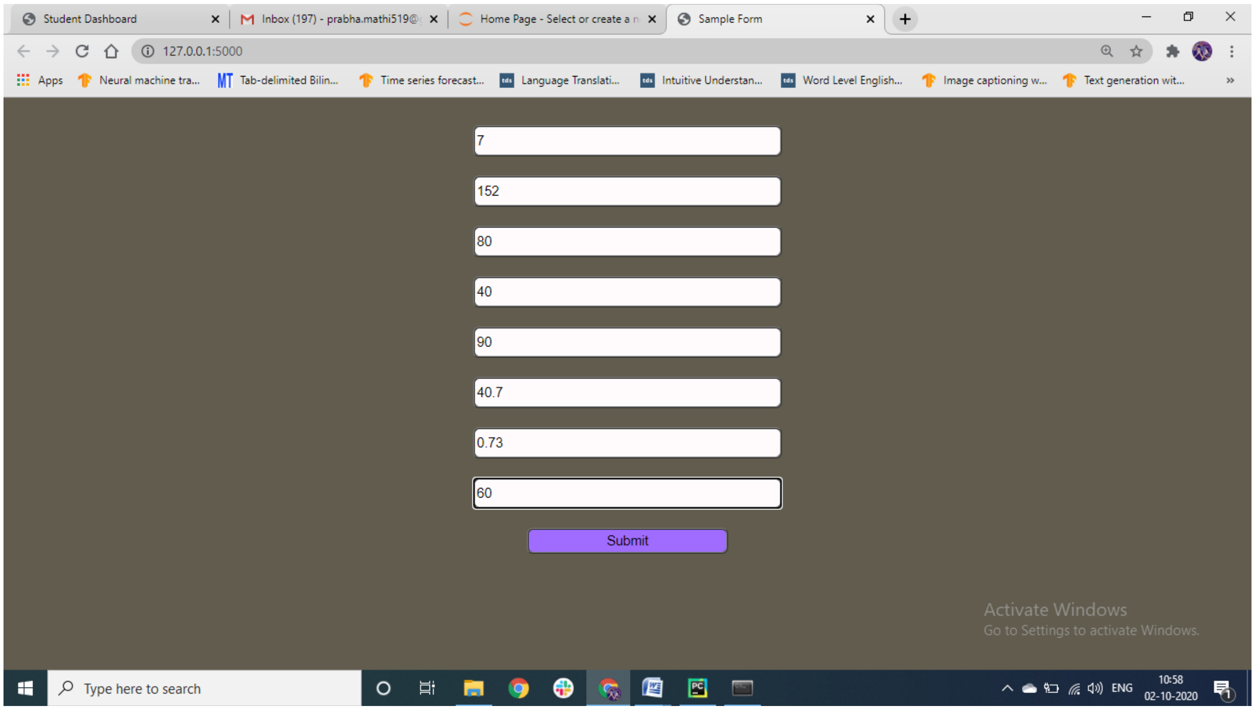
Api Gateway:

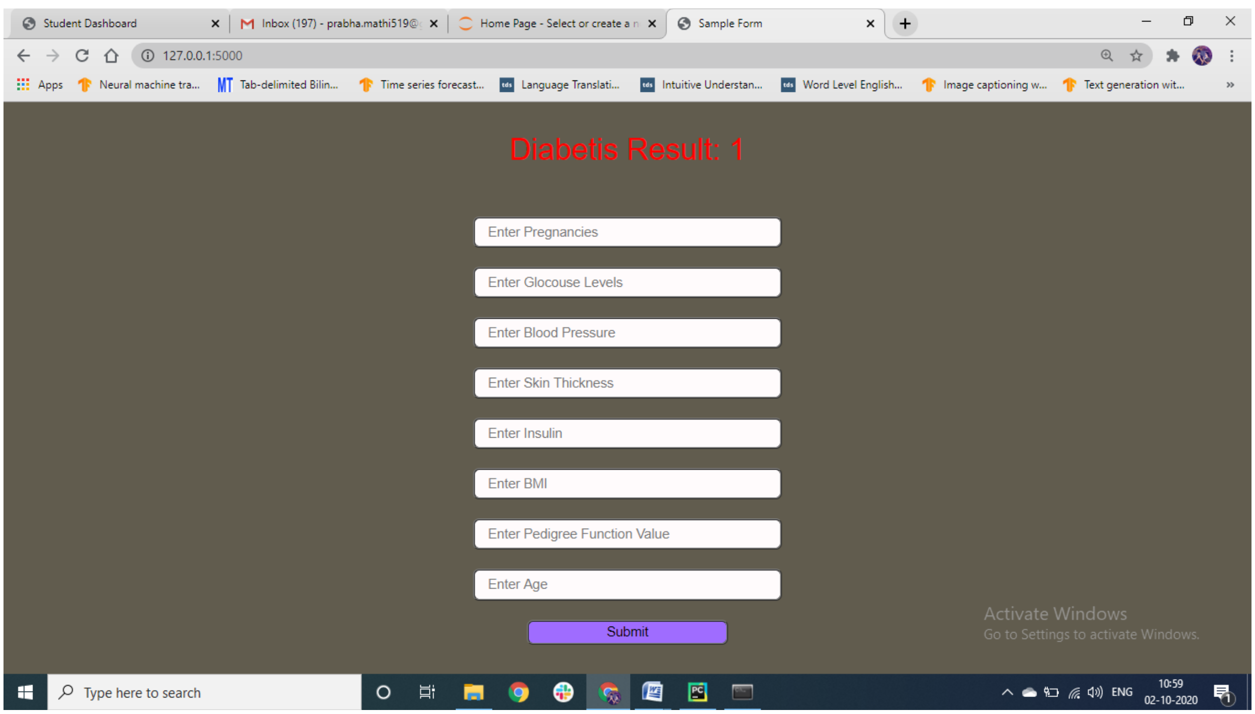
Amazon API Gateway is an AWS service for creating, publishing, maintaining, monitoring, and securing REST, HTTP, and WebSocket APIs at any scale. API developers can create APIs that access AWS or other web services, as well as data stored in the AWS Cloud . API Gateway creates RESTful APIs that: Are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

5.FLOWCHART:



6.RESULT:





7.ADVANTAGES :

1. It helps to detect the diabetics in the earlier stages

2. It helps to easily detects the diabetic

8.APPLICATIONS:

1. Used in early prediction for diabetes for women.
2. We can also use it for predicting heart disease, phenomena, kidney disease
3. We can also use it to predict the medical health condition of the people.

9.FUTURE SCOPE:

This research study has only targeted patients with diabetes. Readmission prediction model needs to be generated for other key health conditions also such as Heart disease, Phenomena, kidney disease etc. in Indian Healthcare system. In the future studies, planned and unplanned(emergency) readmission needs to be considered.

10.CONCLUSION:

Machine learning has the great ability to revolutionize the diabetes risk prediction with the help of advanced computational methods and availability of large amount of epidemiological and genetic diabetes risk dataset. Detection of diabetes in its early stages is the key for treatment. This work has described a machine learning approach to predicting diabetes levels. The technique may also help researchers to develop an accurate and effective tool that will reach at the table of clinicians to help them make better decision about the disease status.

11.BIBILOGRAPHY:

1. Komi, Zhai. 2017. Application of DataMining Methods in Diabetes Prediction

Code:

```python
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

```python
dataset=pd.read_csv('diabetes.csv')
dataset.head()
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1

```python
dataset.shape
```

```
(768, 9)
```

```python
dataset.info()
```

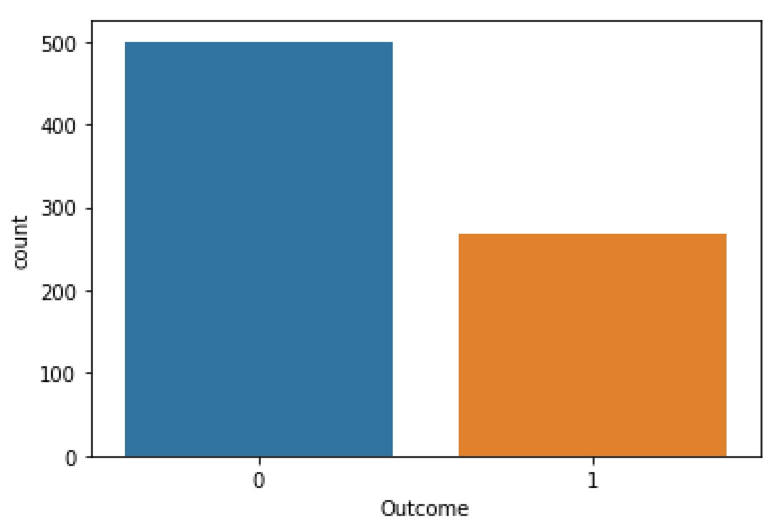
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Pregnancies               768 non-null    int64
 1   Glucose                   768 non-null    int64
 2   BloodPressure             768 non-null    int64
 3   SkinThickness             768 non-null    int64
 4   Insulin                   768 non-null    int64
 5   BMI                       768 non-null    float64
 6   DiabetesPedigreeFunction  768 non-null    float64
 7   Age                       768 non-null    int64
 8   Outcome                   768 non-null    int64
dtypes: float64(2), int64(7)
memory usage: 54.1 KB
```

```python
dataset.describe()
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
count	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000
mean	3.845052	120.894531	69.105469	20.536458	79.799479	31.992578	0.471876	33.240885	0.348958
std	3.369578	31.972618	19.355807	15.952218	115.244002	7.884160	0.331329	11.760232	0.476951
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.078000	21.000000	0.000000
25%	1.000000	99.000000	62.000000	0.000000	0.000000	27.300000	0.243750	24.000000	0.000000
50%	3.000000	117.000000	72.000000	23.000000	30.500000	32.000000	0.372500	29.000000	0.000000
75%	6.000000	140.250000	80.000000	32.000000	127.250000	36.600000	0.626250	41.000000	1.000000
max	17.000000	199.000000	122.000000	99.000000	846.000000	67.100000	2.420000	81.000000	1.000000

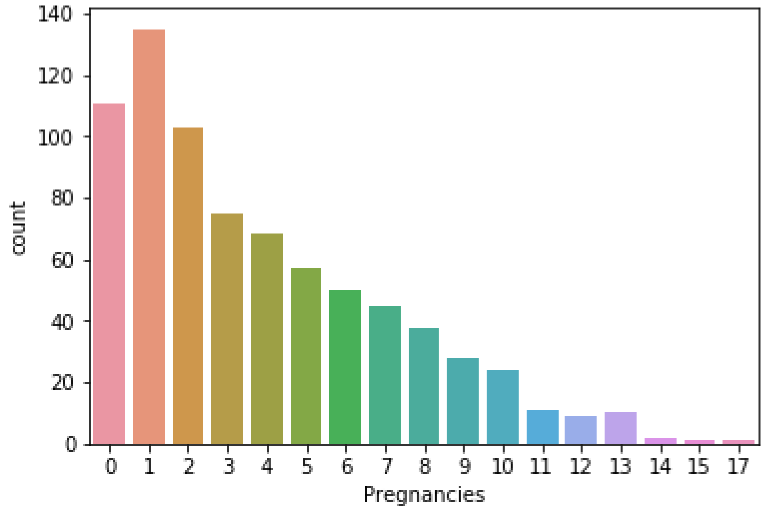
```python
sns.countplot(x='Outcome',data=dataset)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f64f34657b8>
```




```python
sns.countplot(x='Pregnancies',data=dataset)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f64f2c05f98>
```

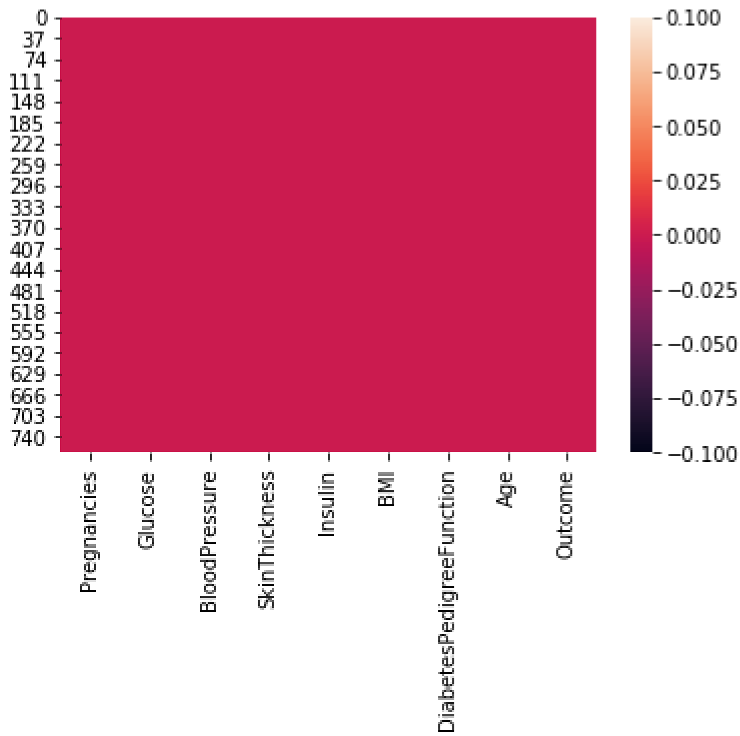



```
1 keys=dataset.keys()[:-1]
2 dataset.isnull().sum()
```

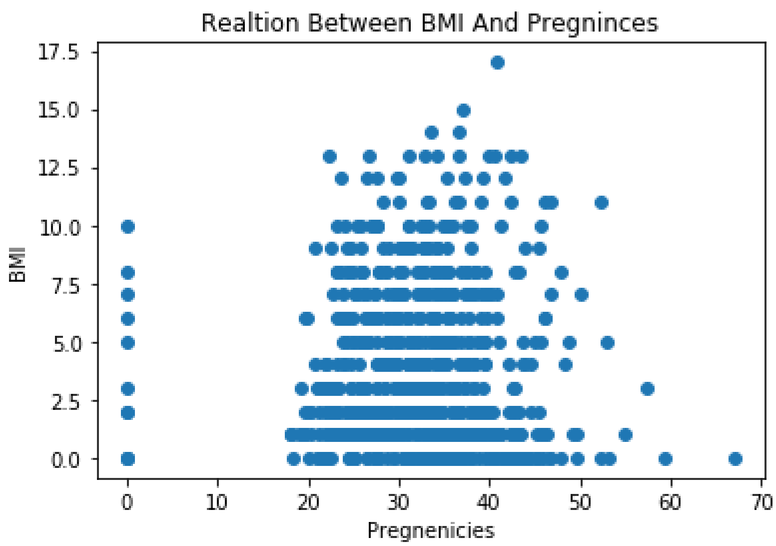
```
Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

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

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f64efdb2048>
```

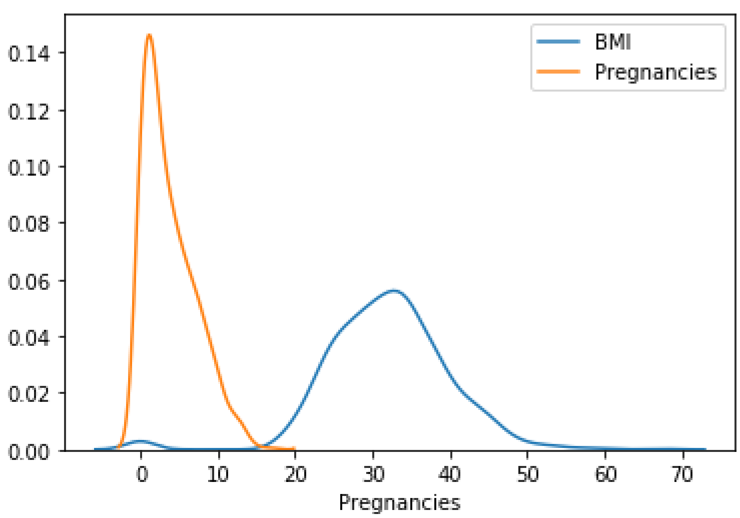



```
plt.scatter(dataset['BMI'],dataset['Pregnancies'])
plt.xlabel('Pregnenicies')
plt.ylabel('BMI')
plt.title('Realtion Between BMI And Pregninces')
plt.show()
```




```
sns.distplot(dataset['BMI'],hist=False,label='BMI')
sns.distplot(dataset['Pregnancies'],hist=False,label='Pregnancies')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f64eeb3dcc0>
```




```
data_in=dataset.iloc[:,:-1]
data_in=np.array(data_in)
data_out=dataset.iloc[:,-1]
```

```
from sklearn.preprocessing import MinMaxScaler
```

```
sc=MinMaxScaler(feature_range=(0,1))
data_in=sc.fit_transform(data_in)
dici={}
for i in range(len(keys)):
    dici.update({keys[i]:data_in[:,i]})
dataset=pd.DataFrame(dici)
dataset.head()
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age
0	0.352941	0.743719	0.590164	0.353535	0.000000	0.500745	0.234415	0.483333
1	0.058824	0.427136	0.540984	0.292929	0.000000	0.396423	0.116567	0.166667
2	0.470588	0.919598	0.524590	0.000000	0.000000	0.347243	0.253629	0.183333
3	0.058824	0.447236	0.540984	0.232323	0.111111	0.418778	0.038002	0.000000
4	0.000000	0.688442	0.327869	0.353535	0.198582	0.642325	0.943638	0.200000

```
final_data=pd.concat([data_out,dataset],axis=1)
final_data.head()
```

	Outcome	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age
0	1	0.352941	0.743719	0.590164	0.353535	0.000000	0.500745	0.234415	0.483333
1	0	0.058824	0.427136	0.540984	0.292929	0.000000	0.396423	0.116567	0.166667
2	1	0.470588	0.919598	0.524590	0.000000	0.000000	0.347243	0.253629	0.183333
3	0	0.058824	0.447236	0.540984	0.232323	0.111111	0.418778	0.038002	0.000000
4	1	0.000000	0.688442	0.327869	0.353535	0.198582	0.642325	0.943638	0.200000

```
from sklearn.model_selection import train_test_split
import boto3,re,os,json,sagemaker
from sagemaker import get_execution_role
```

```
train,test=train_test_split(final_data,test_size=0.2)
role=get_execution_role()
my_region=boto3.session.Session().region_name
```

```
containers = {'us-west-2':
```

```
'433757028032.dkr.ecr.us-west-2.amazonaws.com/xgboost:latest',
              'us-east-1': '811284229777.dkr.ecr.us-east-1.amazonaws.com/xgboost:latest',
              'us-east-2': '825641698319.dkr.ecr.us-east-2.amazonaws.com/xgboost:latest',
              'eu-west-1': '685385470294.dkr.ecr.eu-west-1.amazonaws.com/xgboost:latest'}
```

```
prefix='sagemaker/Diabetis'
bucket_name='buildathonproject1'
```

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

```
sess=sagemaker.Session()
diabetis_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)
diabetis_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)
```

```
diabetis_model.fit({'train':s3_input_train})
```

```
2020-09-29 10:55:12 Starting - Starting the training job...
2020-09-29 10:55:14 Starting - Launching requested ML instances......
2020-09-29 10:56:37 Starting - Preparing the instances for training......
2020-09-29 10:57:19 Downloading - Downloading input data...
2020-09-29 10:58:14 Training - Training image download completed. Training in progress.
2020-09-29 10:58:14 Uploading - Uploading generated training model
2020-09-29 10:58:14 Completed - Training job completed
Arguments: train
[2020-09-29:10:58:01:INFO] Running standalone xgboost training.
[2020-09-29:10:58:01:INFO] Path /opt/ml/input/data/validation does not exist!
[2020-09-29:10:58:01:INFO] File size need to be processed in the node: 0.1mb. Available memory size in the node: 179.14mb
[2020-09-29:10:58:01:INFO] Determined delimiter of CSV input is ','
[10:58:01] S3DistributionType set as FullyReplicated
[10:58:01] 768x8 matrix with 6144 entries loaded from /opt/ml/input/data/train?format=csv&label_column=0&delimiter=,
[10:58:01] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 18 extra nodes, 6 pruned nodes, max_depth=5
[0]#011train-error:0.213542
[10:58:01] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 18 extra nodes, 6 pruned nodes, max_depth=5
[1]#011train-error:0.204427
[10:58:01] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 12 extra nodes, 14 pruned nodes, max_depth=4
```

```python
detector=diabetis_model.deploy(initial_instance_count=1,instance_type='ml.m5.large')
```

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

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

```python
detector.endpoint
```

```
'xgboost-2020-09-29-10-55-12-525'
```

```python
from sagemaker.predictor import csv_serializer
test_data_array=test.drop('Outcome',axis=1).values #load the data into an array
detector.content_type = 'text/csv' # set the data type for an inference
detector.serializer = csv_serializer # set the serializer type
predictions=detector.predict(test_data_array).decode('utf-8') # predict!
predictions_array = np.fromstring(predictions[1:], sep=',')
print(predictions)
```

0.869257152081,0.893998146057,0.381067067385,0.273157775402,0.0283411908895,0.00982582382858,0.373198151588,0.739119589329,0.06
31111264229,0.123311661184,0.307913601398,0.0203247666359,0.629984736443,0.143767222762,0.0814729258418,0.450724750757,0.024638
6341751,0.114363595843,0.110667638481,0.957688510418,0.316517740488,0.0341696403921,0.104469493032,0.645296514034,0.03501924872
4,0.437700629234,0.235520675778,0.042892113328,0.0265930593014,0.0634284541011,0.228885501623,0.0145683744922,0.756480336189,0.
694737255573,0.885303080082,0.132718250155,0.921021819115,0.0263476632535,0.952018260956,0.0719527080655,0.114961370826,0.55295
1216698,0.0137457912788,0.744290709496,0.544826328754,0.587123095989,0.528581261635,0.162170022726,0.0176852233708,0.1297354400
16,0.0428186766803,0.909726798534,0.596715211868,0.173101589084,0.566160440445,0.0577708743513,0.0111106587574,0.159157410264,
0.50178951025,0.0285898968577,0.101399920881,0.0282326750457,0.369609743357,0.711481392384,0.527515113354,0.326200067997,0.1132
53474236,0.0272240731865,0.0114788562059,0.172703176737,0.812383890152,0.0954049006104,0.0128487339243,0.373979181051,0.5377048
85006,0.016701227054,0.0842607021332,0.0228234268725,0.958100438118,0.0116031290963,0.0561438687146,0.0251012574881,0.170562088
49,0.0290512945503,0.503959953785,0.10793094337,0.758869111538,0.293403834105,0.0270585417747,0.914609193802,0.404713571072,0.2
85631388426,0.937867879868,0.498330116272,0.860977768898,0.438471287489,0.0125098237768,0.0745606943965,0.238392338157,0.024982
0854515,0.426272720098,0.0541459918022,0.847022116184,0.399285793304,0.583398222923,0.0148843647912,0.818740963936,0.0431824102
998,0.485654085875,0.0323955453932,0.202002897859,0.159274235368,0.863941371441,0.100214712322,0.0569356866181,0.144378244877,
0.0103294588625,0.034358240664,0.0401456132531,0.355897545815,0.496356070042,0.0171386990696,0.796175658703,0.176519811153,0.06
74795359373,0.0123462602496,0.163044512272,0.211845159531,0.0240886937827,0.171182587743,0.211568906903,0.025868050754 1,0.02155
1316604,0.0357983671129,0.132661700249,0.962401032448,0.225732207298,0.0140418512747,0.614252388477,0.0873962789774,0.672396600
246,0.018287261948,0.20697632432,0.126642733812,0.0169984512031,0.281665205956,0.388219565153,0.0152490651235,0.241831704974,0.
13600166142,0.337226092815,0.887941598892,0.144347906113,0.695236980915

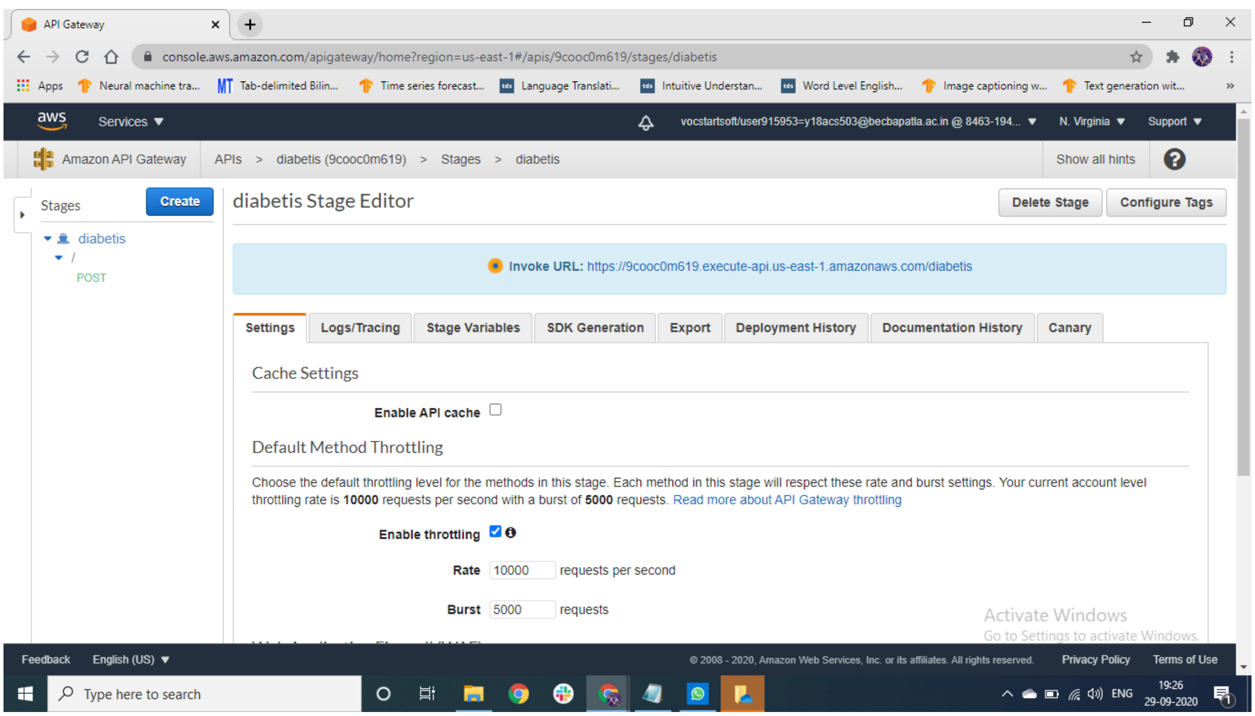
LambdaCode:

```
import os
import io
import boto3
import json
import csv
def lambda_handler(event, context):
    ENDPOINT_NAME = os.environ['envirornment_variable']
    runtime= boto3.client('runtime.sagemaker')
    print(ENDPOINT_NAME)
    print("Received event: " , json.dumps(event, indent=2))
    data = json.loads(json.dumps(event))
    print("Data:",data)
    payload = data['data']
    print("Payload:",payload)
    response = runtime.invoke_endpoint(EndpointName=ENDPOINT_NAME,
                                       ContentType='text/csv',
                                       Body=payload)
    print(response)
    result = json.loads(response['Body'].read().decode())
    print(result)

    if result>0.5:
```

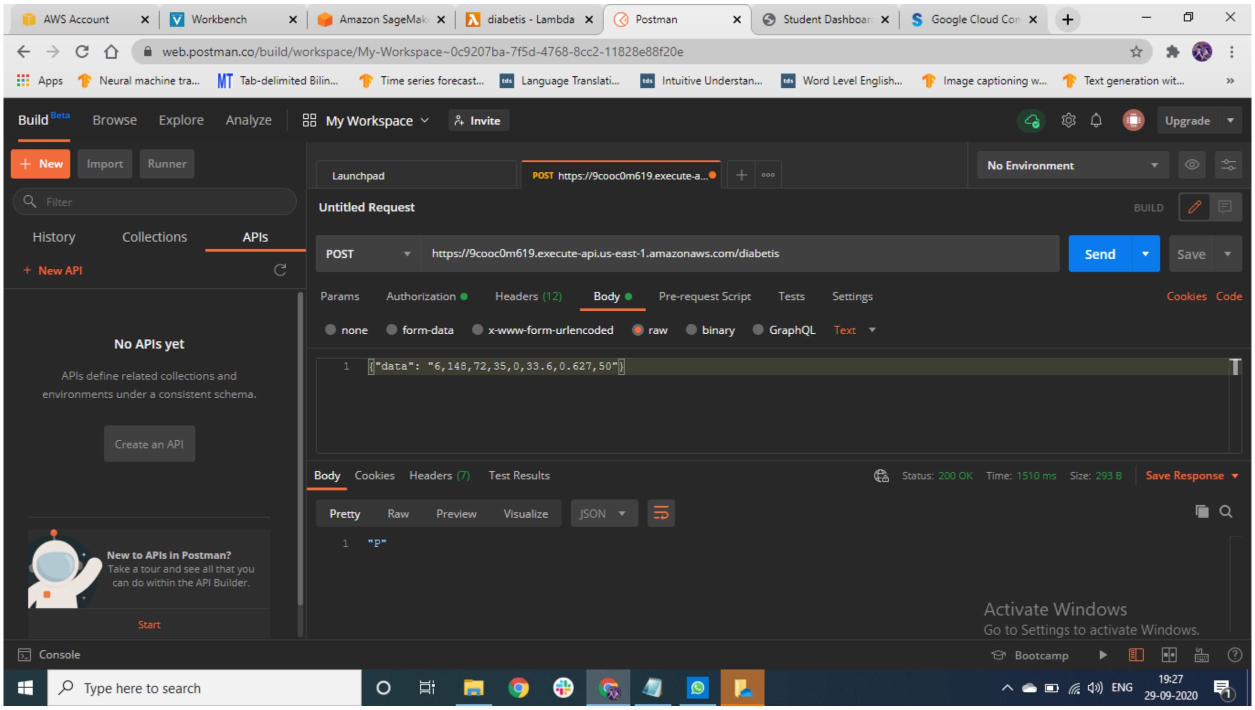
```
        return "P"
    else:
        return "N"

```

```
Response:
"P"

Request ID:
"aa7a54be-8d8b-470e-b1e8-f9aeac7a7ef1"

Function logs:
START RequestId: aa7a54be-8d8b-470e-b1e8-f9aeac7a7ef1 Version: $LATEST
xgboost-2020-09-29-10-55-12-525
Received event:  {
  "data": "6,148,72,35,0,33.6,0.627,50"
}
Data: {'data': '6,148,72,35,0,33.6,0.627,50'}
Payload: 6,148,72,35,0,33.6,0.627,50
{'ResponseMetadata': {'RequestId': '254f60ab-644b-4503-81fe-71c536f06b68', 'HTTPStatusCode': 200, 'HTTPHeaders': {'x-amzn-requestid': '254f60
0.843269765377
```

RESTApiCreation:



TestApi:


Untitled Request
POST
https://9cooc0m619.execute-api.us-east-1.amazonaws.com/diabetis
Send
{"data": "6,148,72,35,0,33.6,0.627,50"}
Status: 200 OK Time: 1510 ms Size: 293 B
"P"