Name: Dhanashri Jagadale

Batch Number: LISUM16

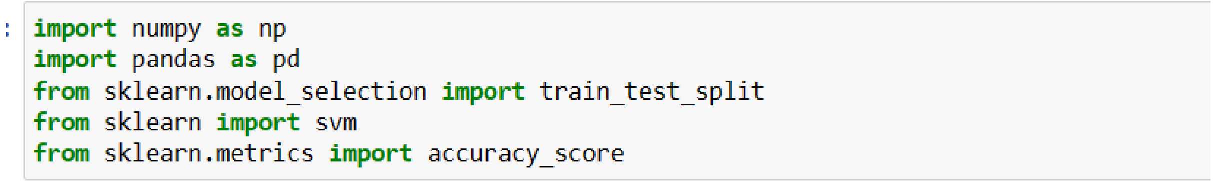
Submission Date: 11/03/2023

Submitted URL:

Deployment on Azure

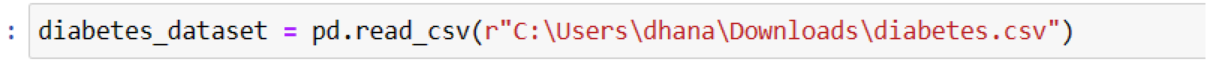
For Diabetics

Step 1: Import the necessary python libraries to run the model.

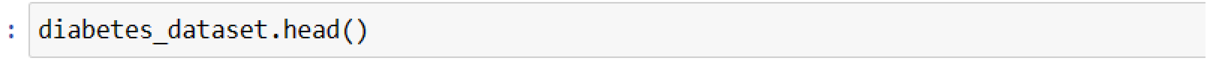


```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn import svm
from sklearn.metrics import accuracy_score
```

Step 2: Reading the dataset

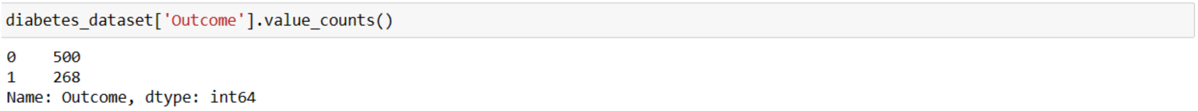


```
diabetes_dataset = pd.read_csv(r"C:\Users\dhana\Downloads\diabetes.csv")
```



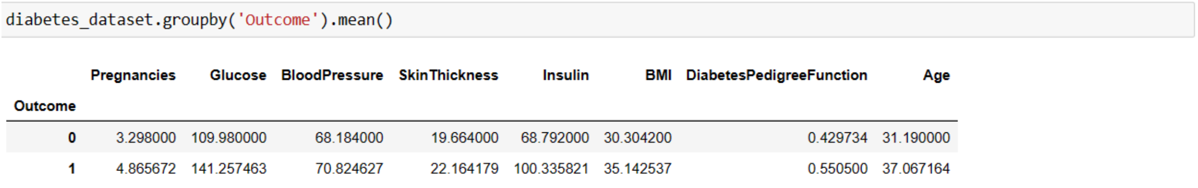
```
diabetes_dataset.head()
```

Step 3: Creating X and Y array



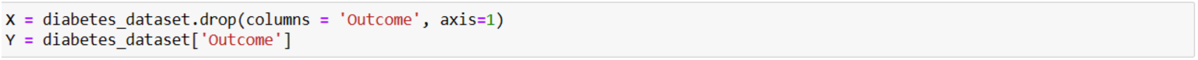
```
diabetes_dataset['Outcome'].value_counts()
```

```
0    500
1    268
Name: Outcome, dtype: int64
```



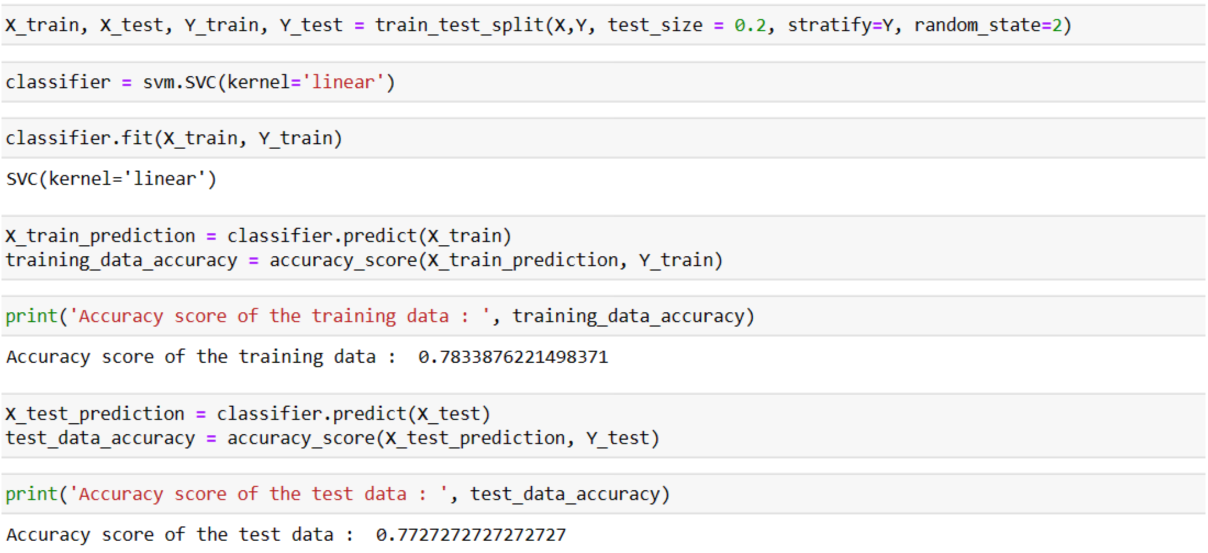
```
diabetes_dataset.groupby('Outcome').mean()
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age
Outcome								
0	3.298000	109.980000	68.184000	19.664000	68.792000	30.304200	0.429734	31.190000
1	4.865672	141.257463	70.824627	22.164179	100.335821	35.142537	0.550500	37.067164



```
X = diabetes_dataset.drop(columns = 'Outcome', axis=1)
Y = diabetes_dataset['Outcome']
```

Step 4: Training the machine learning model



```
X_train, X_test, Y_train, Y_test = train_test_split(X,Y, test_size = 0.2, stratify=Y, random_state=2)
```

```
classifier = svm.SVC(kernel='linear')
```

```
classifier.fit(X_train, Y_train)
```

```
SVC(kernel='linear')
```

```
X_train_prediction = classifier.predict(X_train)
training_data_accuracy = accuracy_score(X_train_prediction, Y_train)
```

```
print('Accuracy score of the training data : ', training_data_accuracy)
```

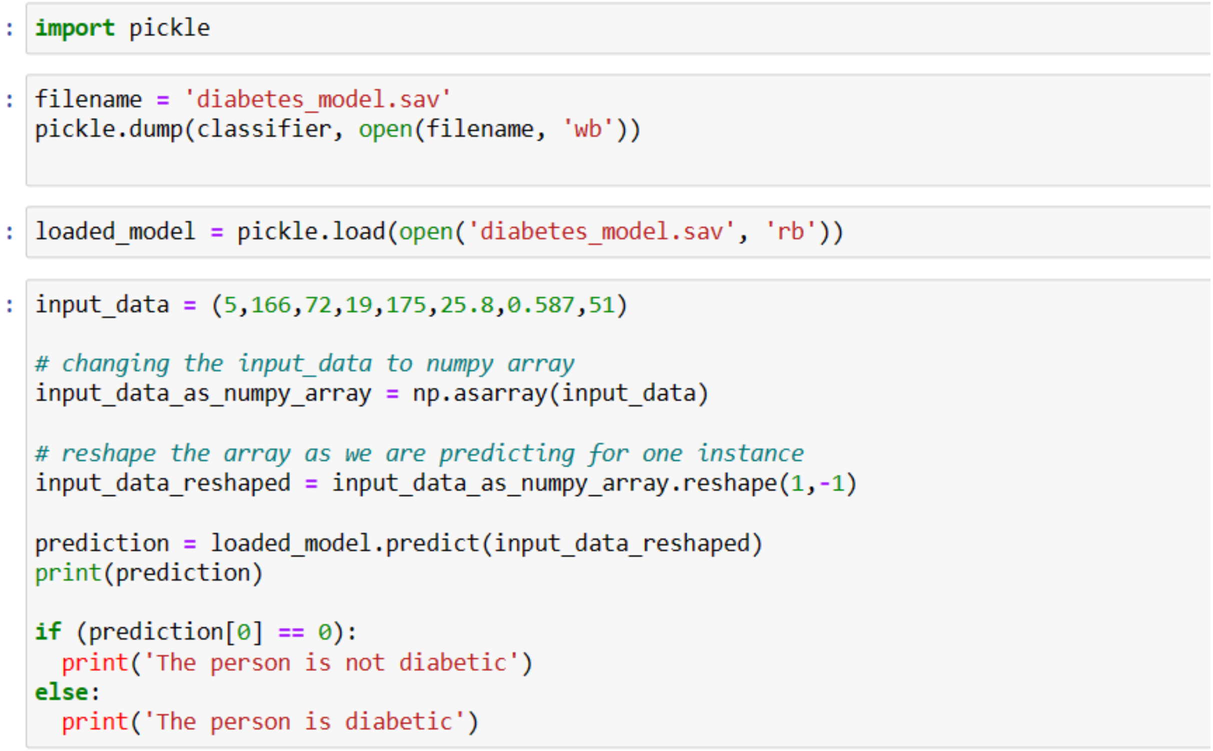
```
Accuracy score of the training data :  0.7833876221498371
```

```
X_test_prediction = classifier.predict(X_test)
test_data_accuracy = accuracy_score(X_test_prediction, Y_test)
```

```
print('Accuracy score of the test data : ', test_data_accuracy)
```

```
Accuracy score of the test data :  0.7727272727272727
```

Step 5: Saving the model using pickle library



```
: import pickle
```

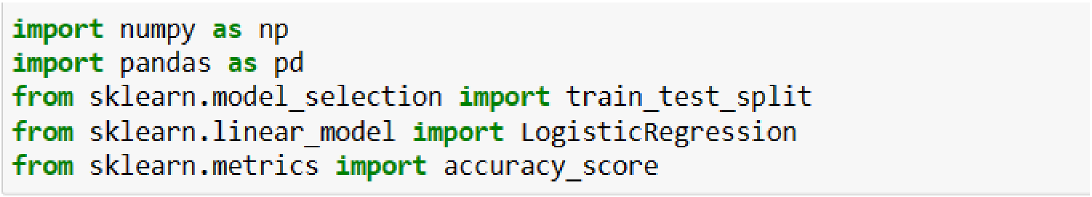
```
: filename = 'diabetes_model.sav'
  pickle.dump(classifier, open(filename, 'wb'))
```

```
: loaded_model = pickle.load(open('diabetes_model.sav', 'rb'))
```

```
: input_data = (5,166,72,19,175,25.8,0.587,51)

  # changing the input_data to numpy array
  input_data_as_numpy_array = np.asarray(input_data)

  # reshape the array as we are predicting for one instance
  input_data_reshaped = input_data_as_numpy_array.reshape(1,-1)

  prediction = loaded_model.predict(input_data_reshaped)
  print(prediction)

  if (prediction[0] == 0):
    print('The person is not diabetic')
  else:
    print('The person is diabetic')
```

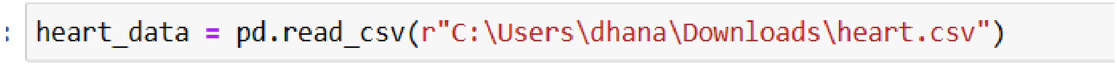
For heart disease

Step 1: Import the necessary python libraries to run the model.

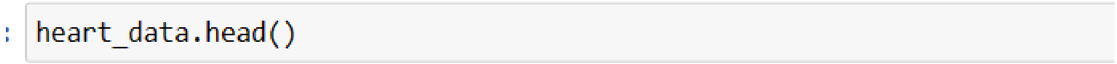


```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
```

Step 2: Reading the dataset

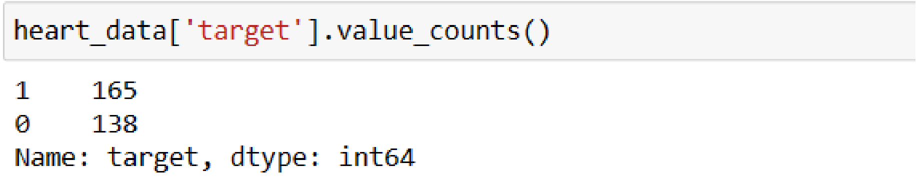


```
heart_data = pd.read_csv(r"C:\Users\dhana\Downloads\heart.csv")
```



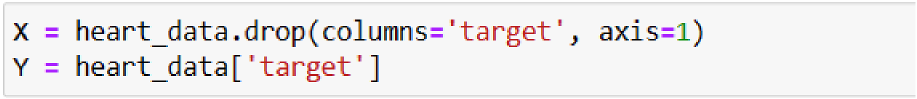
```
heart_data.head()
```

Step 3: Creating X and Y array



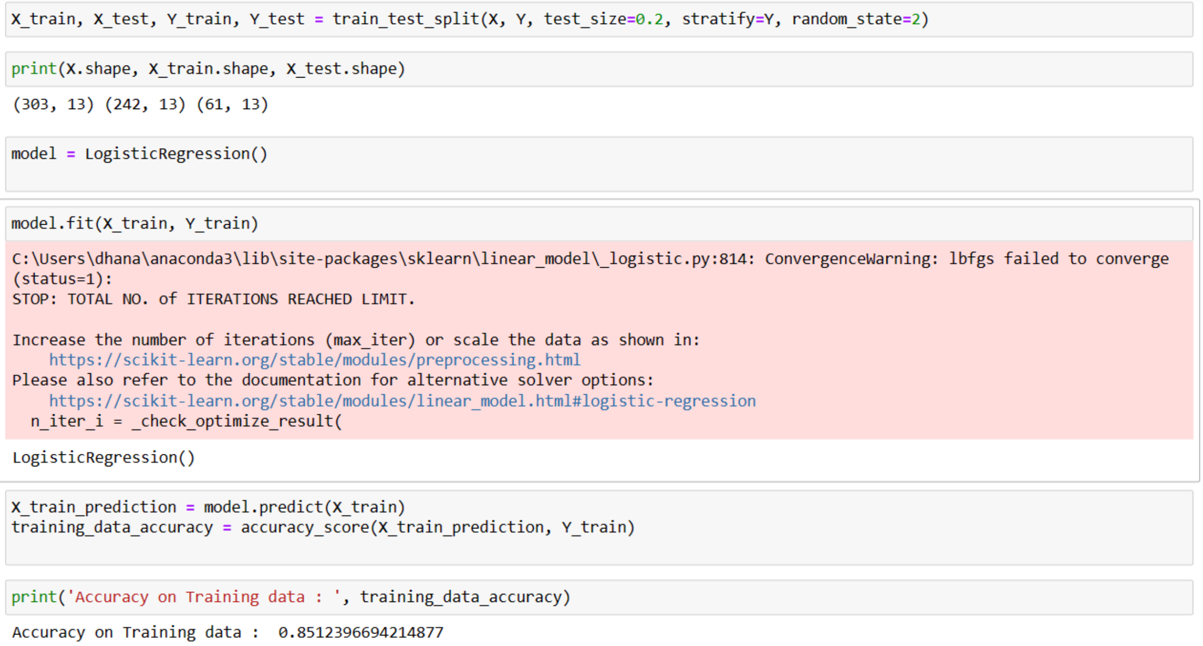
```
heart_data['target'].value_counts()
```

```
1    165
0    138
Name: target, dtype: int64
```



```
X = heart_data.drop(columns='target', axis=1)
Y = heart_data['target']
```

Step 4: Training the machine learning model



```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, stratify=Y, random_state=2)
```

```
print(X.shape, X_train.shape, X_test.shape)
```

```
(303, 13) (242, 13) (61, 13)
```

```
model = LogisticRegression()
```

```
model.fit(X_train, Y_train)
```

```
C:\Users\dhana\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:814: ConvergenceWarning: lbfgs failed to converge
(status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

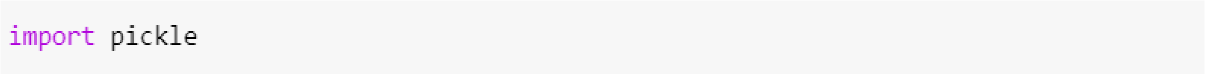
```
LogisticRegression()
```

```
X_train_prediction = model.predict(X_train)
training_data_accuracy = accuracy_score(X_train_prediction, Y_train)
```

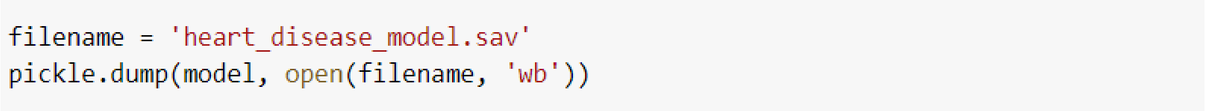
```
print('Accuracy on Training data : ', training_data_accuracy)
```

```
Accuracy on Training data :  0.8512396694214877
```

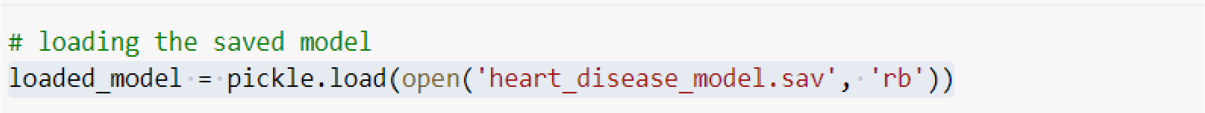
Step 5: Saving the model using pickle library



```
import pickle
```



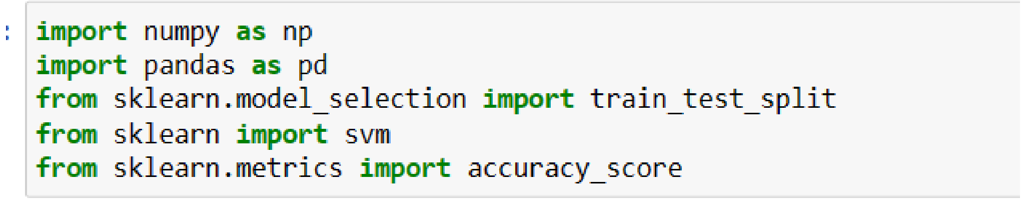
```
filename = 'heart_disease_model.sav'
pickle.dump(model, open(filename, 'wb'))
```



```
# loading the saved model
loaded_model = pickle.load(open('heart_disease_model.sav', 'rb'))
```

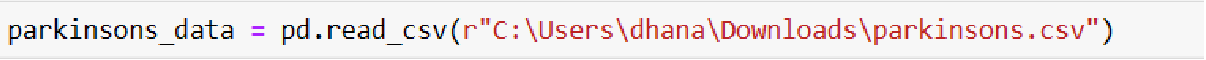
For heart disease

Step 1: Import the necessary python libraries to run the model.

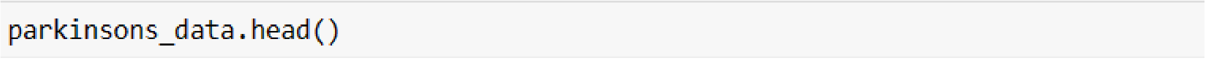


```
:  import numpy as np
   import pandas as pd
   from sklearn.model_selection import train_test_split
   from sklearn import svm
   from sklearn.metrics import accuracy_score
```

Step 2: Reading the dataset

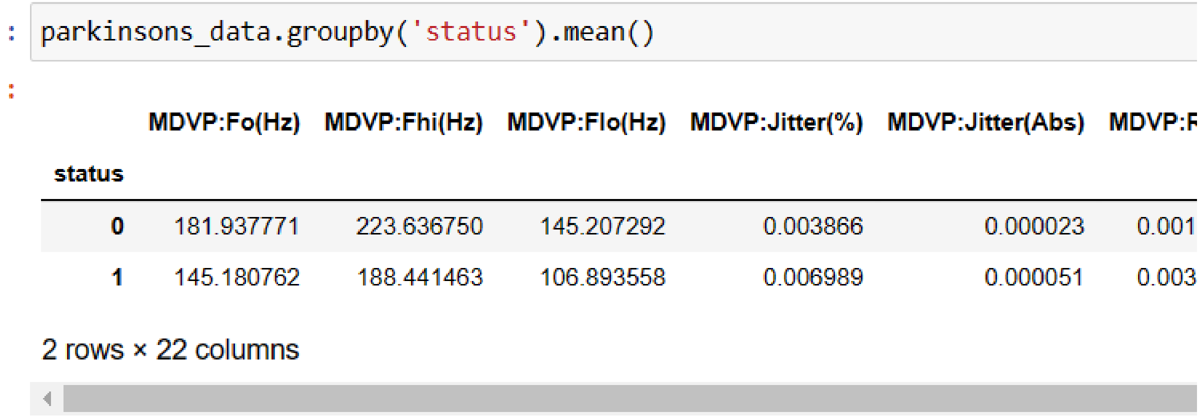


```
parkinsons_data = pd.read_csv(r"C:\Users\dhana\Downloads\parkinsons.csv")
```



```
parkinsons_data.head()
```

Step 3: Creating X and Y array

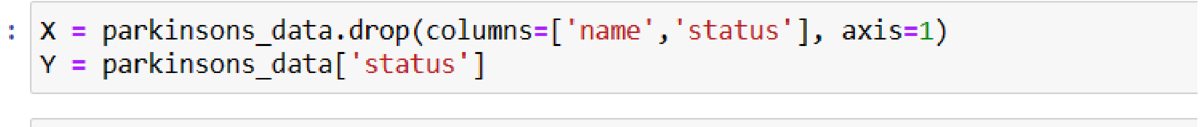


```
:  parkinsons_data.groupby('status').mean()
```

:

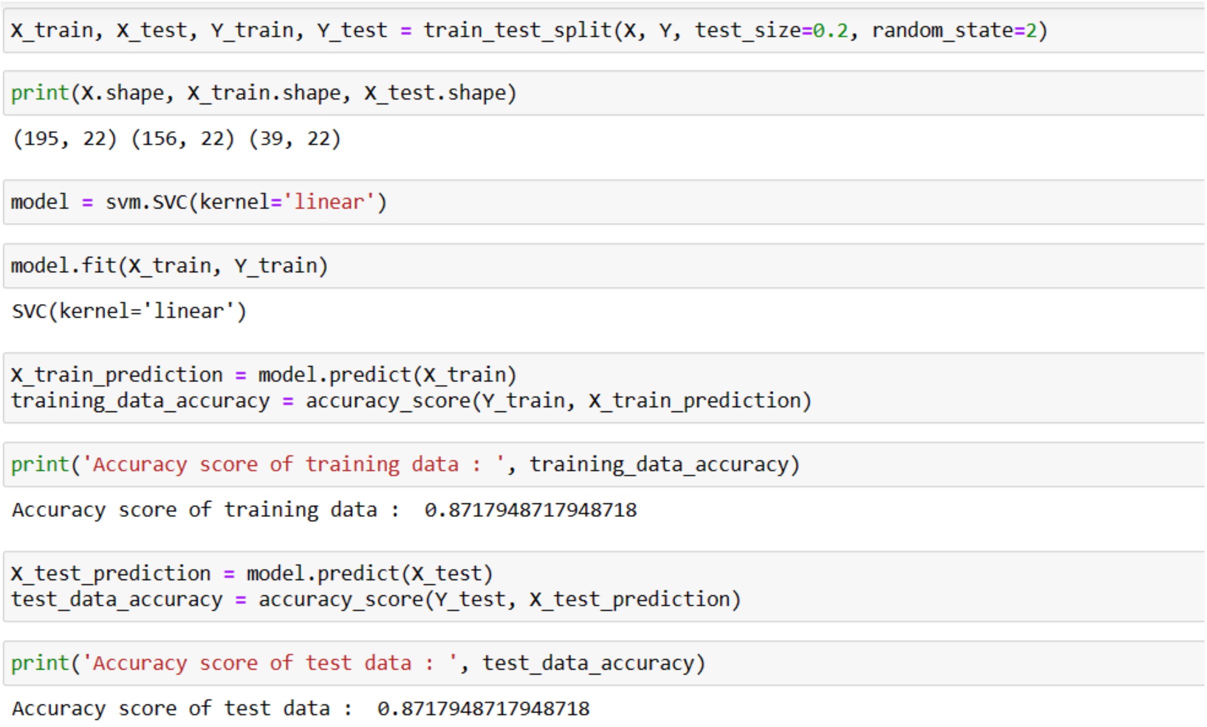
	MDVP:Fo(Hz)	MDVP:Fhi(Hz)	MDVP:Flo(Hz)	MDVP:Jitter(%)	MDVP:Jitter(Abs)	MDVP:F
status						
0	181.937771	223.636750	145.207292	0.003866	0.000023	0.001
1	145.180762	188.441463	106.893558	0.006989	0.000051	0.003

2 rows × 22 columns



```
:  X = parkinsons_data.drop(columns=['name','status'], axis=1)
   Y = parkinsons_data['status']
```

Step 4: Training the machine learning model



```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_state=2)
```

```
print(X.shape, X_train.shape, X_test.shape)
```

```
(195, 22) (156, 22) (39, 22)
```

```
model = svm.SVC(kernel='linear')
```

```
model.fit(X_train, Y_train)
```

```
SVC(kernel='linear')
```

```
X_train_prediction = model.predict(X_train)
training_data_accuracy = accuracy_score(Y_train, X_train_prediction)
```

```
print('Accuracy score of training data : ', training_data_accuracy)
```

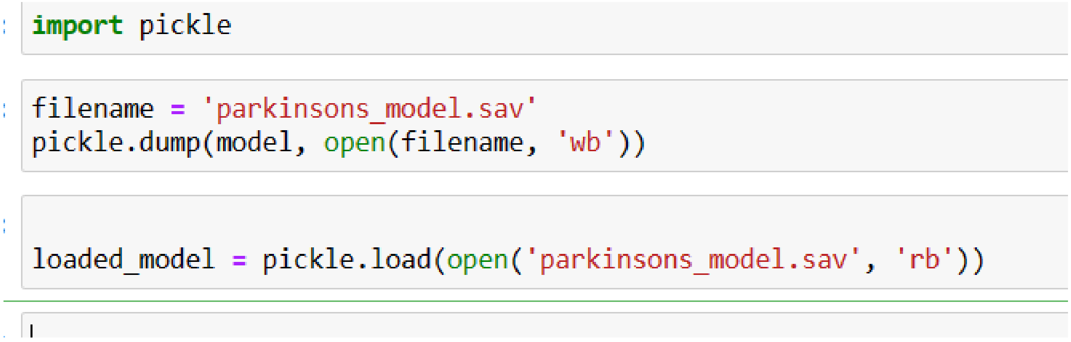
```
Accuracy score of training data :  0.8717948717948718
```

```
X_test_prediction = model.predict(X_test)
test_data_accuracy = accuracy_score(Y_test, X_test_prediction)
```

```
print('Accuracy score of test data : ', test_data_accuracy)
```

```
Accuracy score of test data :  0.8717948717948718
```

Step 5: Saving the model using pickle library



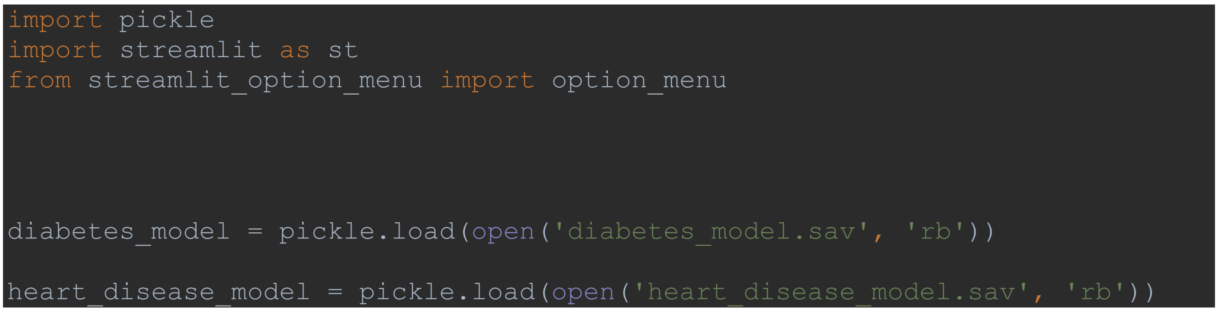
```
import pickle
```

```
filename = 'parkinsons_model.sav'
pickle.dump(model, open(filename, 'wb'))
```

```

loaded_model = pickle.load(open('parkinsons_model.sav', 'rb'))
```

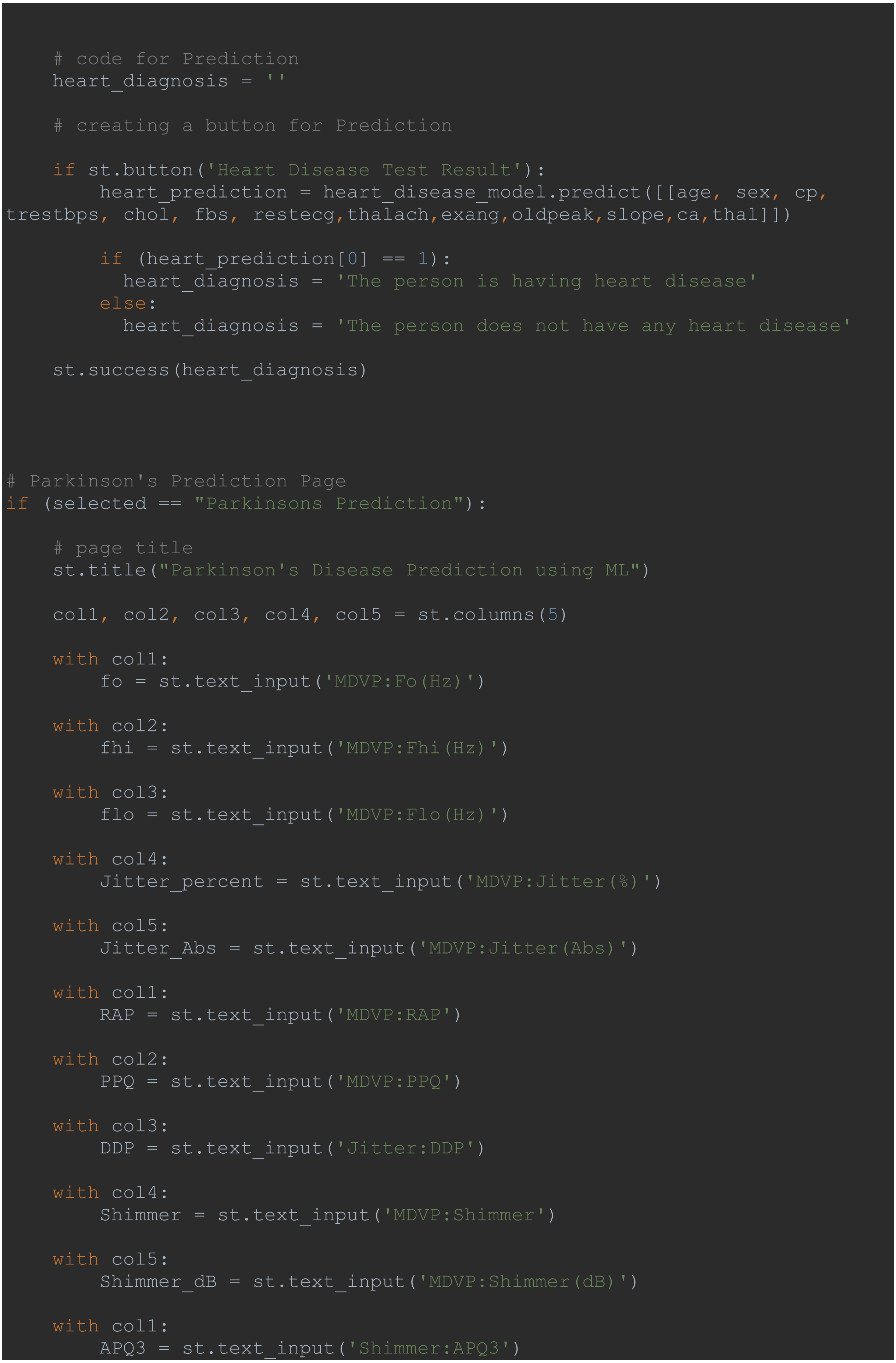
Step 6: Deployment of the model on heroku

The app.py file using streamlit



```
import pickle
import streamlit as st
from streamlit_option_menu import option_menu




diabetes_model = pickle.load(open('diabetes_model.sav', 'rb'))

heart_disease_model = pickle.load(open('heart_disease_model.sav', 'rb'))
```

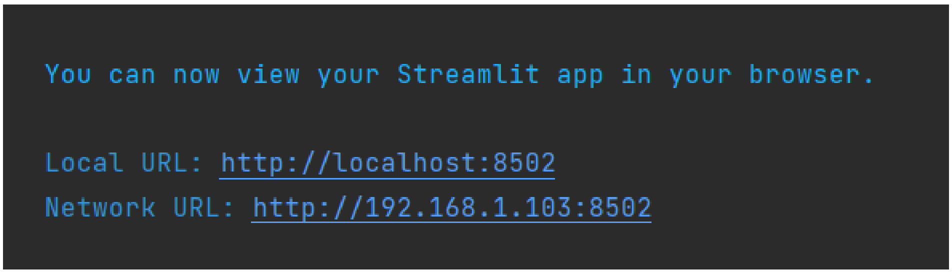
```
parkinsons_model = pickle.load(open('parkinsons_model.sav', 'rb'))




with st.sidebar:

    selected = option_menu('Multiple Disease Prediction System',

                          ['Diabetes Prediction',
                           'Heart Disease Prediction',
                           'Parkinsons Prediction'],
                          icons=['activity','heart','person'],
                          default_index=0)


# Diabetes Prediction Page
if (selected == 'Diabetes Prediction'):

    # page title
    st.title('Diabetes Prediction using ML')


    # getting the input data from the user
    col1, col2, col3 = st.columns(3)

    with col1:
        Pregnancies = st.text_input('Number of Pregnancies')

    with col2:
        Glucose = st.text_input('Glucose Level')

    with col3:
        BloodPressure = st.text_input('Blood Pressure value')

    with col1:
        SkinThickness = st.text_input('Skin Thickness value')

    with col2:
        Insulin = st.text_input('Insulin Level')

    with col3:
        BMI = st.text_input('BMI value')

    with col1:
        DiabetesPedigreeFunction = st.text_input('Diabetes Pedigree
Function value')

    with col2:
        Age = st.text_input('Age of the Person')


    # code for Prediction
    diab_diagnosis = ''

    # creating a button for Prediction

    if st.button('Diabetes Test Result'):
        diab_prediction = diabetes_model.predict([[Pregnancies, Glucose,
BloodPressure, SkinThickness, Insulin, BMI, DiabetesPedigreeFunction,
Age]])

        if (diab_prediction[0] == 1):
          diab_diagnosis = 'The person is diabetic'
        else:
          diab_diagnosis = 'The person is not diabetic'

    st.success(diab_diagnosis)



# Heart Disease Prediction Page
if (selected == 'Heart Disease Prediction'):

    # page title
    st.title('Heart Disease Prediction using ML')

    col1, col2, col3 = st.columns(3)

    with col1:
        age = st.text_input('Age')

    with col2:
        sex = st.text_input('Sex')

    with col3:
        cp = st.text_input('Chest Pain types')

    with col1:
        trestbps = st.text_input('Resting Blood Pressure')

    with col2:
        chol = st.text_input('Serum Cholestoral in mg/dl')

    with col3:
        fbs = st.text_input('Fasting Blood Sugar > 120 mg/dl')

    with col1:
        restecg = st.text_input('Resting Electrocardiographic results')

    with col2:
        thalach = st.text_input('Maximum Heart Rate achieved')

    with col3:
        exang = st.text_input('Exercise Induced Angina')

    with col1:
        oldpeak = st.text_input('ST depression induced by exercise')

    with col2:
        slope = st.text_input('Slope of the peak exercise ST segment')

    with col3:
        ca = st.text_input('Major vessels colored by flourosopy')

    with col1:
        thal = st.text_input('thal: 0 = normal; 1 = fixed defect; 2 =
reversable defect')
```



```
    # code for Prediction
    heart_diagnosis = ''

    # creating a button for Prediction

    if st.button('Heart Disease Test Result'):
        heart_prediction = heart_disease_model.predict([[age, sex, cp,
trestbps, chol, fbs, restecg,thalach,exang,oldpeak,slope,ca,thal]])

        if (heart_prediction[0] == 1):
          heart_diagnosis = 'The person is having heart disease'
        else:
          heart_diagnosis = 'The person does not have any heart disease'

    st.success(heart_diagnosis)




# Parkinson's Prediction Page
if (selected == "Parkinsons Prediction"):

    # page title
    st.title("Parkinson's Disease Prediction using ML")

    col1, col2, col3, col4, col5 = st.columns(5)

    with col1:
        fo = st.text_input('MDVP:Fo(Hz)')

    with col2:
        fhi = st.text_input('MDVP:Fhi(Hz)')

    with col3:
        flo = st.text_input('MDVP:Flo(Hz)')

    with col4:
        Jitter_percent = st.text_input('MDVP:Jitter(%)')

    with col5:
        Jitter_Abs = st.text_input('MDVP:Jitter(Abs)')

    with col1:
        RAP = st.text_input('MDVP:RAP')

    with col2:
        PPQ = st.text_input('MDVP:PPQ')

    with col3:
        DDP = st.text_input('Jitter:DDP')

    with col4:
        Shimmer = st.text_input('MDVP:Shimmer')

    with col5:
        Shimmer_dB = st.text_input('MDVP:Shimmer(dB)')

    with col1:
        APQ3 = st.text_input('Shimmer:APQ3')
    with col2:
        APQ5 = st.text_input('Shimmer:APQ5')

    with col3:
        APQ = st.text_input('MDVP:APQ')

    with col4:
        DDA = st.text_input('Shimmer:DDA')

    with col5:
        NHR = st.text_input('NHR')

    with col1:
        HNR = st.text_input('HNR')

    with col2:
        RPDE = st.text_input('RPDE')

    with col3:
        DFA = st.text_input('DFA')

    with col4:
        spread1 = st.text_input('spread1')

    with col5:
        spread2 = st.text_input('spread2')

    with col1:
        D2 = st.text_input('D2')

    with col2:
        PPE = st.text_input('PPE')



    # code for Prediction
    parkinsons_diagnosis = ''

    # creating a button for Prediction
    if st.button("Parkinson's Test Result"):
        parkinsons_prediction = parkinsons_model.predict([[fo, fhi, flo,
Jitter_percent, Jitter_Abs, RAP,
PPQ,DDP,Shimmer,Shimmer_dB,APQ3,APQ5,APQ,DDA,NHR,HNR,RPDE,DFA,spread1,sprea
d2,D2,PPE]])

        if (parkinsons_prediction[0] == 1):
          parkinsons_diagnosis = "The person has Parkinson's disease"
        else:
          parkinsons_diagnosis = "The person does not have Parkinson's
disease"

    st.success(parkinsons_diagnosis)
```

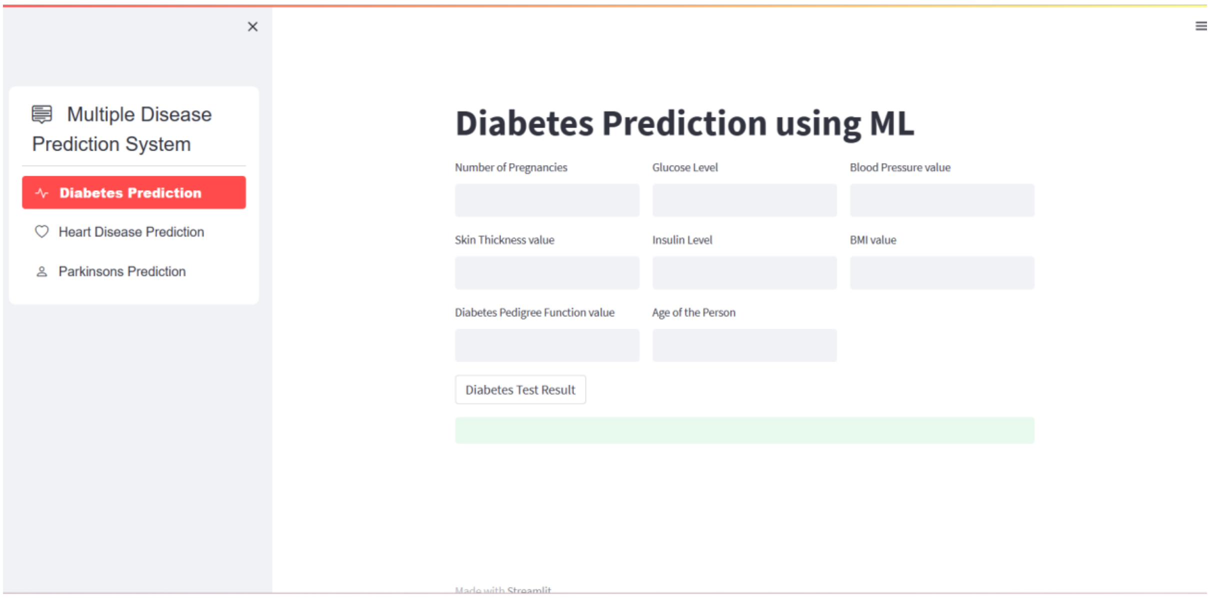
The Html code for Predict page

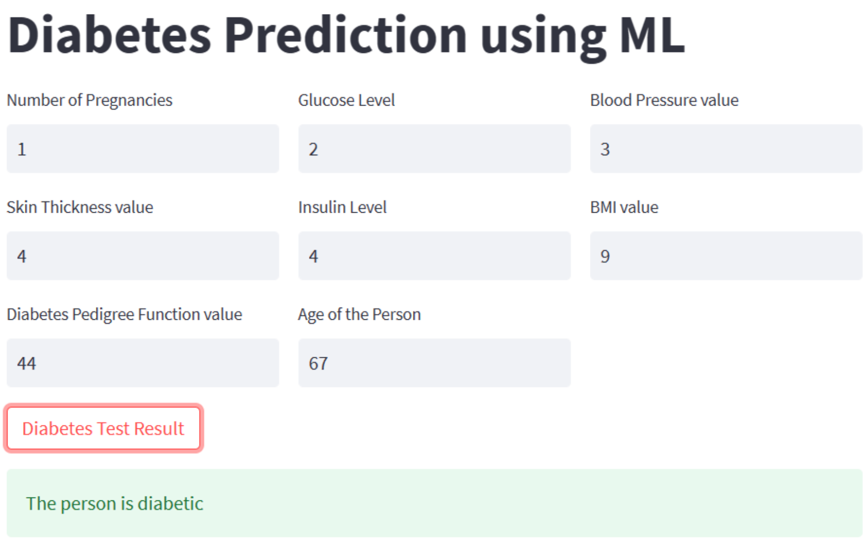
Step 6: Result after running



```
You can now view your Streamlit app in your browser.

Local URL: http://localhost:8502
Network URL: http://192.168.1.103:8502
```

D





Diabetes Prediction using ML

Number of Pregnancies	Glucose Level	Blood Pressure value
1	2	3

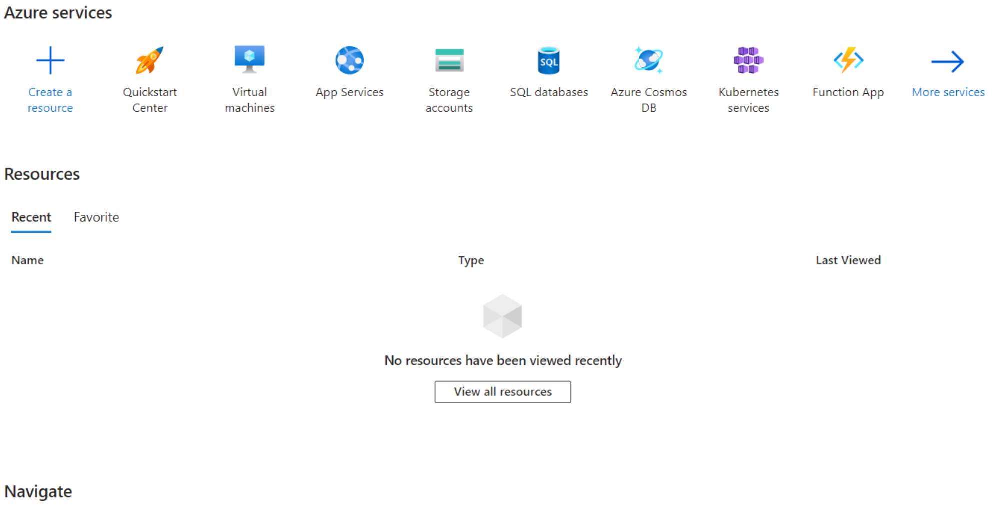
Skin Thickness value	Insulin Level	BMI value
4	4	9

Diabetes Pedigree Function value	Age of the Person
44	67

The person is diabetic

Azure

1) Select create a resource



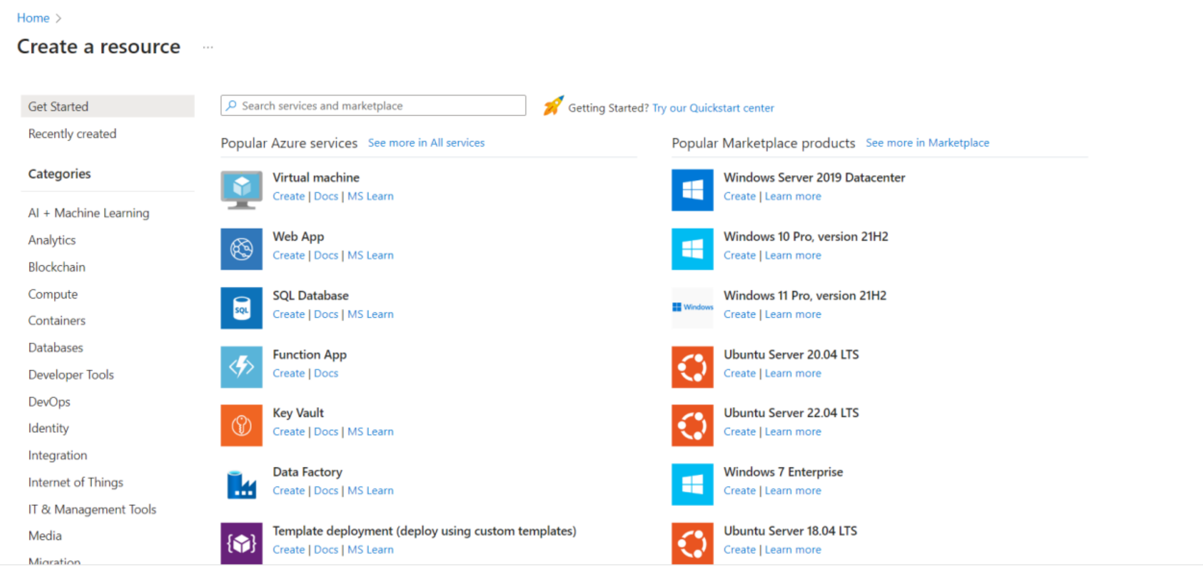
Azure services

Resources

No resources have been viewed recently

Navigate

2)Select create web app



Home >

Create a resource ···

- Get Started
- Recently created

Categories

- AI + Machine Learning
- Analytics
- Blockchain
- Compute
- Containers
- Databases
- Developer Tools
- DevOps
- Identity
- Integration
- Internet of Things
- IT & Management Tools
- Media
- Migration

Search services and marketplace

Getting Started? Try our Quickstart center

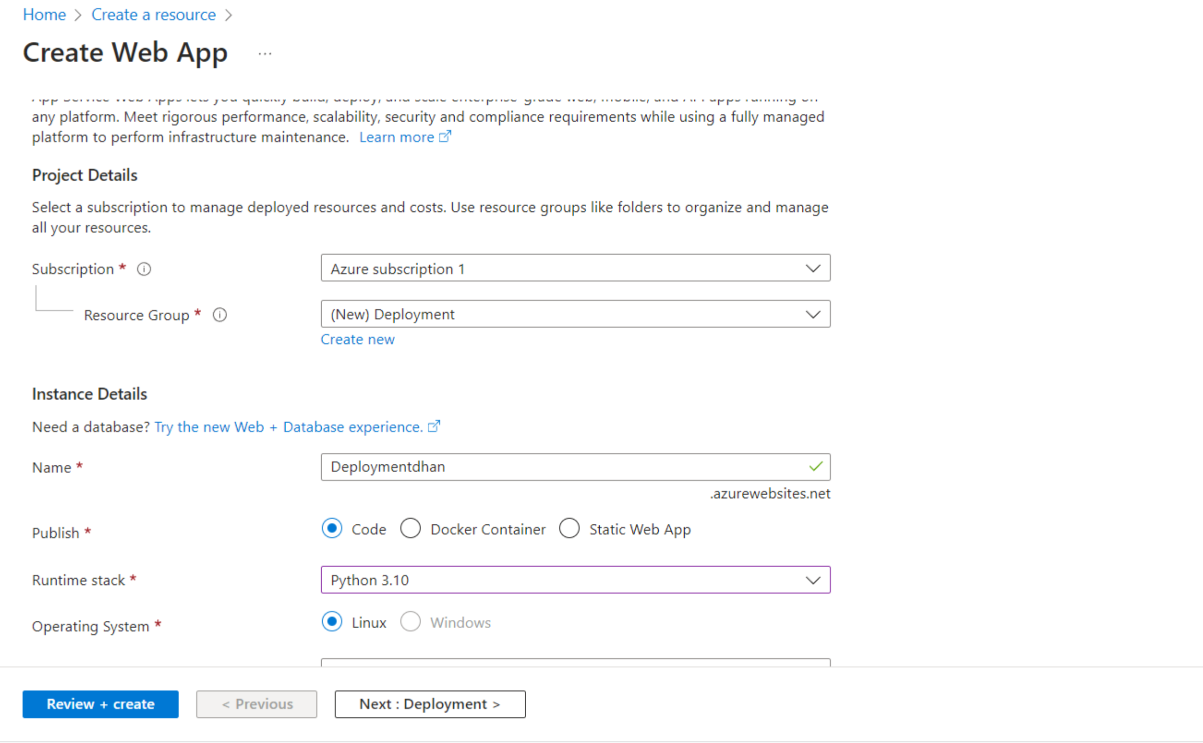
Popular Azure services See more in All services

- Virtual machine — Create | Docs | MS Learn
- Web App — Create | Docs | MS Learn
- SQL Database — Create | Docs | MS Learn
- Function App — Create | Docs
- Key Vault — Create | Docs | MS Learn
- Data Factory — Create | Docs | MS Learn
- Template deployment (deploy using custom templates) — Create | Docs | MS Learn

Popular Marketplace products See more in Marketplace

- Windows Server 2019 Datacenter — Create | Learn more
- Windows 10 Pro, version 21H2 — Create | Learn more
- Windows 11 Pro, version 21H2 — Create | Learn more
- Ubuntu Server 20.04 LTS — Create | Learn more
- Ubuntu Server 22.04 LTS — Create | Learn more
- Windows 7 Enterprise — Create | Learn more
- Ubuntu Server 18.04 LTS — Create | Learn more

3)Fill the require information about the app



Home > Create a resource >

Create Web App ···

App Service Web Apps lets you quickly build, deploy, and scale enterprise-grade web, mobile, and API apps running on any platform. Meet rigorous performance, scalability, security and compliance requirements while using a fully managed platform to perform infrastructure maintenance. Learn more

Project Details

Select a subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription * : Azure subscription 1

Resource Group * : (New) Deployment

Create new

Instance Details

Need a database? Try the new Web + Database experience.

Name * : Deploymentdhan ✓

.azurewebsites.net

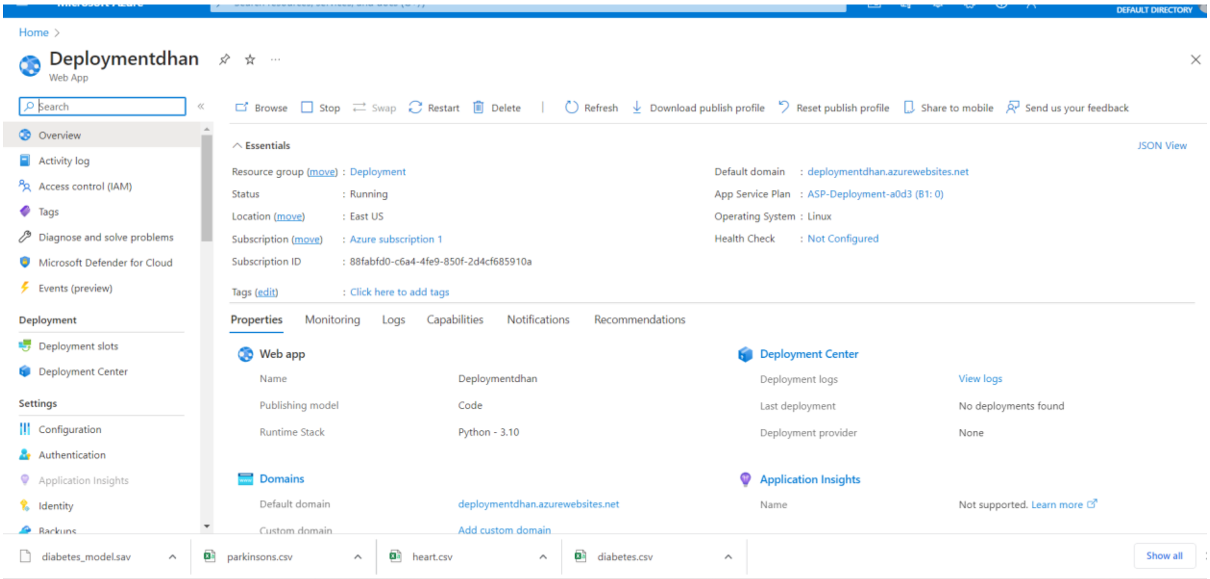
Publish * : ● Code ○ Docker Container ○ Static Web App

Runtime stack * : Python 3.10

Operating System * : ● Linux ○ Windows

Review + create | < Previous | Next : Deployment >

4)Click on Deployment Centre



Essentials

Resource group (move) : Deployment
Status : Running
Location (move) : East US
Subscription (move) : Azure subscription 1
Subscription ID : 88fabfd0-c6a4-4fe9-850f-2d4cf685910a
Tags (edit) : Click here to add tags

Default domain : deploymentdhan.azurewebsites.net
App Service Plan : ASP-Deployment-a0d3 (B1: 0)
Operating System : Linux
Health Check : Not Configured

Web app

Name	Deploymentdhan
Publishing model	Code
Runtime Stack	Python - 3.10

Deployment Center

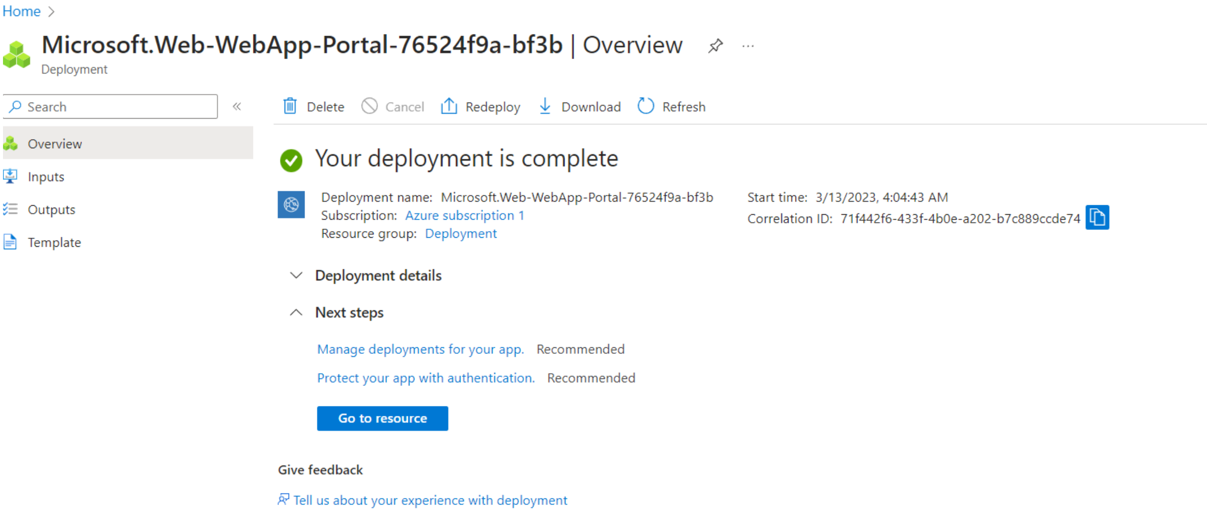
Deployment logs	View logs
Last deployment	No deployments found
Deployment provider	None

Domains

Default domain	deploymentdhan.azurewebsites.net
Custom domain	Add custom domain

Application Insights

Name	Not supported. Learn more



Your deployment is complete

Deployment name: Microsoft.Web-WebApp-Portal-76524f9a-bf3b
Subscription: Azure subscription 1
Resource group: Deployment
Start time: 3/13/2023, 4:04:43 AM
Correlation ID: 71f442f6-433f-4b0e-a202-b7c889ccde74

Deployment details

Next steps

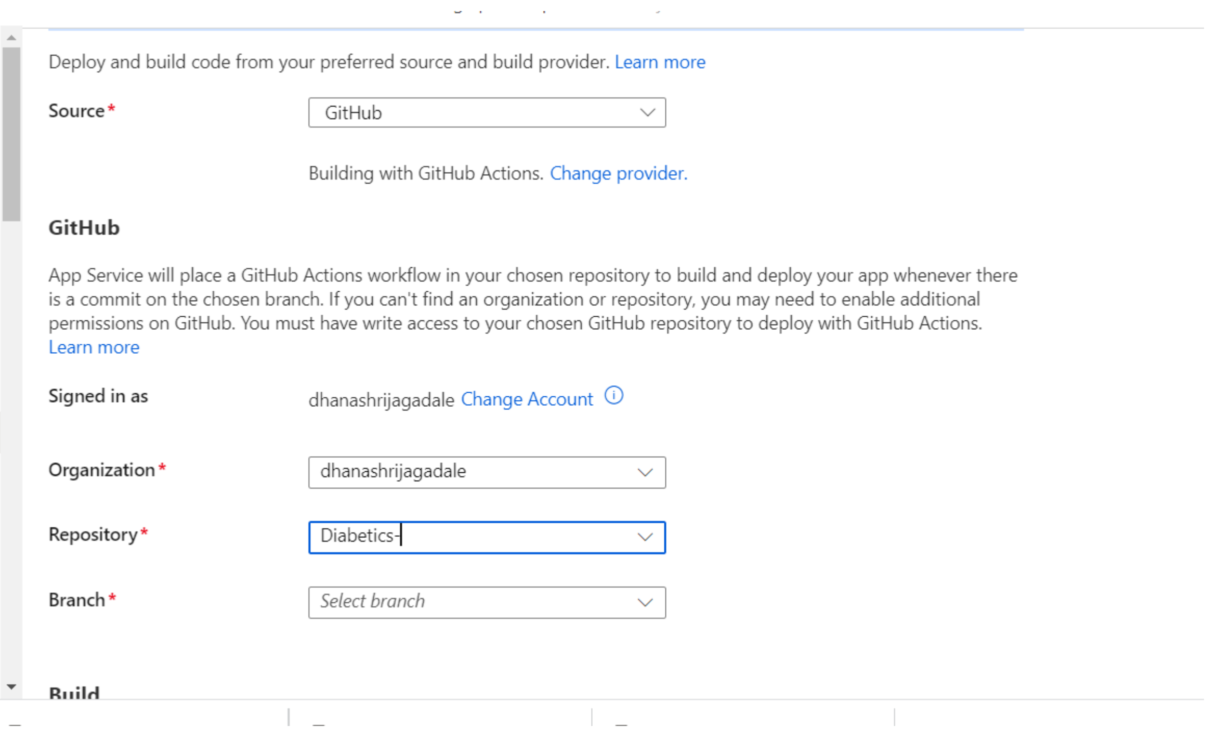
Manage deployments for your app. Recommended

Protect your app with authentication. Recommended

Give feedback

Tell us about your experience with deployment

5) Deploying the code through github



Deploy and build code from your preferred source and build provider. Learn more

Source * GitHub

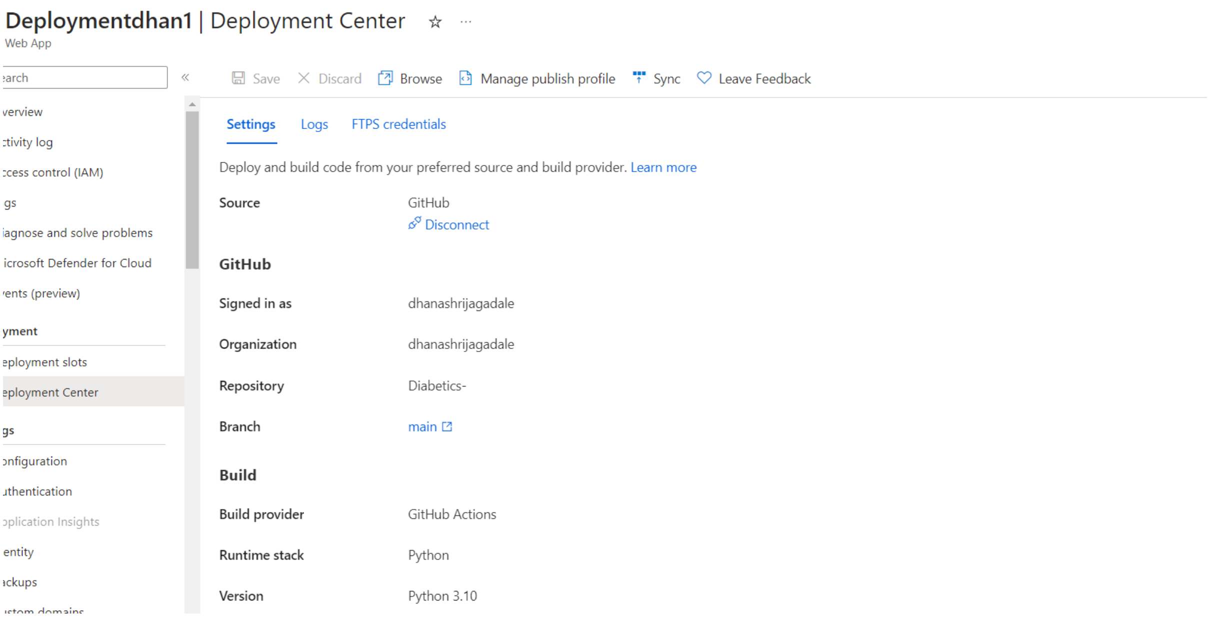
Building with GitHub Actions. Change provider.

GitHub

App Service will place a GitHub Actions workflow in your chosen repository to build and deploy your app whenever there is a commit on the chosen branch. If you can't find an organization or repository, you may need to enable additional permissions on GitHub. You must have write access to your chosen GitHub repository to deploy with GitHub Actions. Learn more

Signed in as	dhanashrijagadale Change Account ⓘ
Organization *	dhanashrijagadale
Repository *	Diabetics-
Branch *	*Select branch*

Build



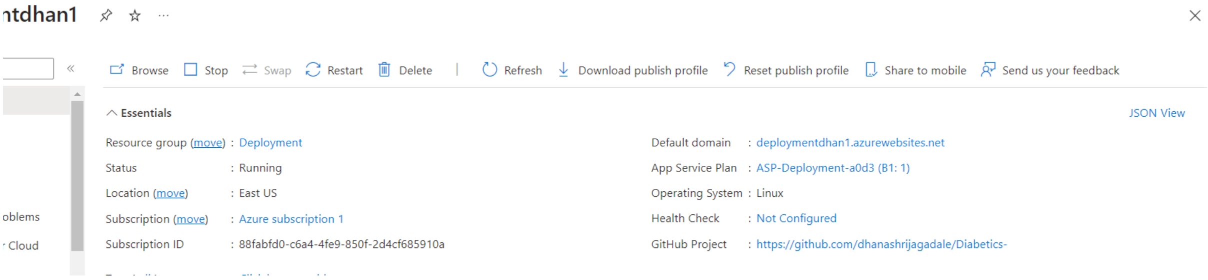
Deploymentdhan1 | Deployment Center ☆ ···

Web App

Deploy and build code from your preferred source and build provider. Learn more

Source	GitHub Disconnect
GitHub	
Signed in as	dhanashrijagadale
Organization	dhanashrijagadale
Repository	Diabetics-
Branch	main
Build	
Build provider	GitHub Actions
Runtime stack	Python
Version	Python 3.10

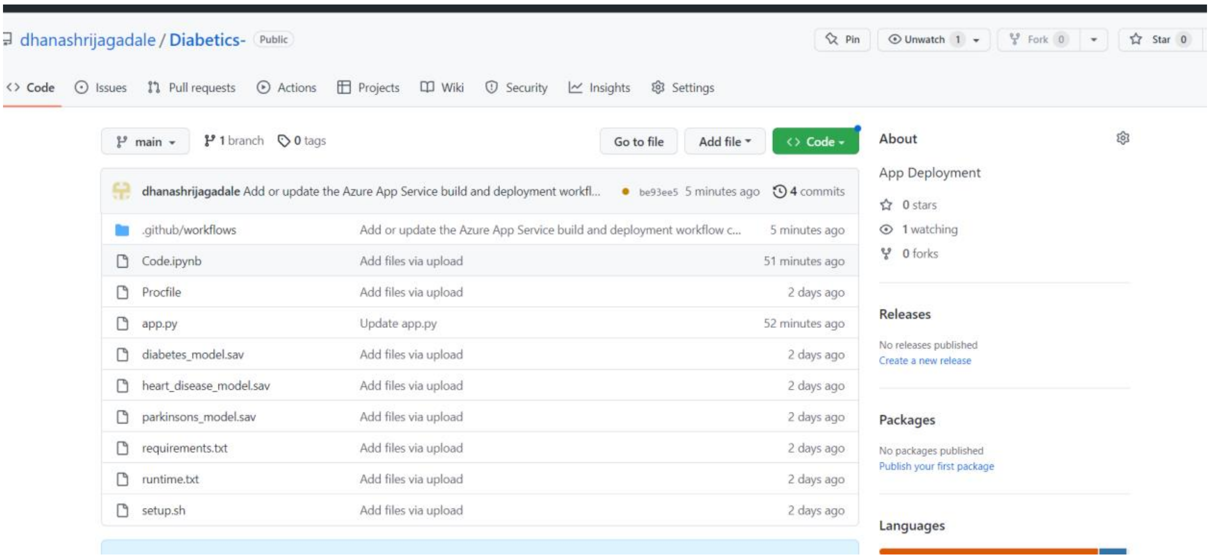
Done



ntdhan1

Essentials

Resource group (move)	: Deployment	Default domain	: deploymentdhan1.azurewebsites.net
Status	: Running	App Service Plan	: ASP-Deployment-a0d3 (B1: 1)
Location (move)	: East US	Operating System	: Linux
Subscription (move)	: Azure subscription 1	Health Check	: Not Configured
Subscription ID	: 88fabfd0-c6a4-4fe9-850f-2d4cf685910a	GitHub Project	: https://github.com/dhanashrijagadale/Diabetics-



dhanashrijagadale Add or update the Azure App Service build and deployment workfl... be93ee5 5 minutes ago 4 commits

.github/workflows	Add or update the Azure App Service build and deployment workflow c...	5 minutes ago
Code.ipynb	Add files via upload	51 minutes ago
Procfile	Add files via upload	2 days ago
app.py	Update app.py	52 minutes ago
diabetes_model.sav	Add files via upload	2 days ago
heart_disease_model.sav	Add files via upload	2 days ago
parkinsons_model.sav	Add files via upload	2 days ago
requirements.txt	Add files via upload	2 days ago
runtime.txt	Add files via upload	2 days ago
setup.sh	Add files via upload	2 days ago

About

App Deployment

0 stars

1 watching

0 forks

Releases

No releases published
Create a new release

Packages

No packages published
Publish your first package

Languages