



**Data Glacier**

Your Deep Learning Partner

# Model deployment (Cloud and API)

Virtual Internship

Bilikis O. Alayo

LISUM31

April 5, 2024

<https://github.com/Omolara-Alayo/Data-Science-Internship/tree/main/Week%205/ML%20model%20deployment>

# Dataset

- A simple heart disease dataset consisting of 304 rows and 13 features was used for this study. The data was cleaned and preprocessed for model training.
- The dataset was scaled and split into training set and test set in the ratio of 70:30.

```
In [2]: # Read the heart disease data into a DataFrame
heart_disease_data = pd.read_csv('heart.csv')
```

```
In [3]: # Display the first 5 rows of the dataset
heart_disease_data.head()
```

```
Out[3]:
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	1
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	1
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	1
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	1
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	1

```
# Checking the shape of the dataset
print(("rows, columns"), heart_disease_data.shape)
```

# Model training

- A logistic regression classifier was used for the training and the model was validated and persisted for later use.

## Training the model Using Logistic Regression

```
In [11]: import time
import pickle
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import cross_val_score
from sklearn.metrics import confusion_matrix, classification_report
from sklearn import metrics

In [13]: # Initializing and training the model

start_time = time.time() #Initializing the start time to capture the execution time
logreg = LogisticRegression()
logreg_model = logreg.fit(X_train, y_train)

logreg_time = time.time() - start_time
print('Accuracy of Logistics Regression classifier on training set: {:.2f}'.format(logreg_model.score(X_train, y_train)))
print('Accuracy of Logistics Regression classifier on test set: {:.2f}'.format(logreg_model.score(X_test, y_test)))
print('Logistic Regression classifier Training Time (seconds):', logreg_time)

Accuracy of Logistics Regression classifier on training set: 0.86
Accuracy of Logistics Regression classifier on test set: 0.79
Logistic Regression classifier Training Time (seconds): 0.006406545639038086

In [14]: #predicting heart disease using with the logistics regression model

heart_disease_prediction = logreg_model.predict(X_test)
heart_disease_prediction[:10]

Out[14]: array([1, 1, 1, 0, 1, 1, 0, 1, 1, 1], dtype=int64)

In [15]: # Evaluating the Logistics Regression Classifier using cross validation setting CV = 5.

logreg_accuracy = cross_val_score(logreg_model, X_train, y_train, cv=5)
print("Logistic Regression Accuracy:", round(logreg_accuracy.mean()*100, 2), "%")

Logistic Regression Accuracy: 84.41 %
```

```
[21]: # Persisting the Heart disease predictive model
```

```
import pickle

model_dump = pickle.dumps(logreg_model)
with open('heart_disease_model.pkl', 'wb') as model_file:
    model_file.write(model_dump)
```

```
[23]: # Predicting heart_disease from the model
```

```
#Loading the Logistic Regression model
with open('heart_disease_model.pkl', 'rb') as model_file:
    loaded_model = pickle.load(model_file)

heart_disease_predict = loaded_model.predict(X_test)
print("Heart disease Predictions using the loaded Logistic Regression model:")
print(heart_disease_predict[:10])
```

```
Heart disease Predictions using the loaded Logistic Regression model:
[1 1 1 0 1 1 0 1 1 1]
```

# Flask App

After training the model, a flask app was built for a web application.

```
app.py
C: > Users > berly > OneDrive > Desktop > Data Glacier_intern > Week 5 > New folder > app.py > ...
1  #!/usr/bin/env python
2  # coding: utf-8
3
4  # Flask app for heart predictive model
5
6  Run Cell | Run Below | Debug Cell
7  # In[ ]:
8
9  import numpy as np
10 from flask import Flask, request, render_template
11 import pickle
12
13 app = Flask(__name__)
14
15 #Prediction function
16 def HeartDiseasePredictor(predict_list):
17     to_predict = np.array(predict_list).reshape(1,13)
18     loaded_model = pickle.load(open("heart_disease_model.pkl", "rb"))
19     result = loaded_model.predict(to_predict)
20     return result[0]
21
22 @app.route('/')
23 def home():
24     return render_template('index.html')
```

```
25
26 @app.route('/predict', methods=['POST'])
27 def predict():
28     if request.method == 'POST':
29         predict_list = request.form.to_dict()
30         predict_list = list(predict_list.values())
31         try:
32             predict_list = [int(float(x)) for x in predict_list]
33         except ValueError:
34             return "Invalid input. Please provide valid integer or float values."
35
36         result = HeartDiseasePredictor(predict_list)
37         if int(result) == 1:
38             prediction = 'This person is suffering from heart disease..'
39         else:
40             prediction = 'This person is not suffering from heart disease..'
41         return render_template("index.html", prediction=prediction)
42
43
44 if __name__=="__main__":
45     app.run(port=5000)
46
47
48 Run Cell | Run Above | Debug Cell
49 # In[ ]:
```

# App form

A HTML file was created for the web application form

```
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4    <meta charset="UTF-8">
5    <meta name="viewport" content="width=device-width, initial-scale=1.0">
6    <title>Heart Disease Predictor</title>
7    <style>
8      body {
9        font-family: Arial, sans-serif;
10       background-color: #19dfdef;
11      }
12      h3 {
13        color: #333;
14      }
15      form {
16        background-color: #f5e08bd3;
17        border-radius: 10px;
18        padding: 20px;
19        box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
20        width: 500px;
21        margin: 0 auto;
22      }
23      label {
24        display: block;
25        margin-bottom: 5px;
26        color: #555;
27      }
28      input[type="number"],
29      select {
30        width: calc(100% - 22px);
31        padding: 8px;
32        margin-bottom: 10px;
33        border: 1px solid #ccc;
34        border-radius: 5px;
35      }
36      input[type="submit"] {
37        width: 100%;
38        padding: 10px;
39        background-color: #4CAF50;
40        color: white;
41        border: none;
42        border-radius: 5px;
43        cursor: pointer;
44        font-size: 16px;
45      }
46      input[type="submit"]:hover {
47        background-color: #45a049;
48      }
49    </style>
50  </head>
51  <body>
52    <h1>Heart Disease Predictor</h1>
53    <div>
54      <form action="/predict" method="POST">
55        <label for="age">Age</label>
56        <input type="number" id="age" name="age">
57        <label for="sex">Sex</label>
58        <select id="sex" name="sex">
59          <option value="0">Female</option>
60          <option value="1">Male</option>
61        </select>
62        <label for="cp">Chest Pain Type</label>
63        <select id="cp" name="cp">
64          <option value="0">Angina</option>
65          <option value="1">Pericarditis</option>
66          <option value="2">Myocarditis</option>
67          <option value="3">Cardiomyopathy</option>
68        </select>
```

# App form (contd)

A HTML file was created for the web application form

```
69 <label for="trestbps">Resting blood pressure (in mm Hg on admission to the hospital)</label>100
70 <input type="number" id="trestbps" name="trestbps">101
71 <label for="chol">Serum Cholesterol in mg/dl</label>102
72 <input type="number" id="chol" name="chol">103
73 <label for="fbs">Fasting blood sugar > 120 mg/dl</label>104
74 <select id="fbs" name="fbs">105
75 | <option value="0">False</option>106
76 | <option value="1">True</option>107
77 </select>108
78 <label for="restecg">Resting electrocardiographic measurement</label>109
79 <select id="restecg" name="restecg">110
80 | <option value="0">Normal</option>111
81 | <option value="1">ST-T wave abnormality</option>112
82 | <option value="2">Probable or definite left ventricular hypertrophy</option>113
83 </select>114
84 <label for="thalach">Maximum heart rate achieved</label>115
85 <input type="number" id="thalach" name="thalach">116
86 <label for="exang">Exercise induced angina</label>117
87 <select id="exang" name="exang">118
88 | <option value="0">No</option>119
89 | <option value="1">Yes</option>120
90 </select>121
91 <label for="oldpeak">ST depression induced by exercise relative to rest</label>
92 <input type="number" id="oldpeak" name="oldpeak" step="0.1">
93
94 <label for="slope">The slope of the peak exercise ST segment</label>
95 <select id="slope" name="slope">
96 | <option value="1">Upsloping</option>
97 | <option value="2">Flat</option>
98 | <option value="3">Downsloping</option>
99 </select>

<label for="ca">Number of major vessels (0-3)</label>
<input type="number" id="ca" name="ca">
<label for="thal">Blood disorder called thalassemia</label>
<select id="thal" name="thal">
| <option value="1">Normal</option>
| <option value="2">Fixed defect</option>
| <option value="3">Reversible defect</option>
</select>
<input type="submit" value="Predict">
</form>
<br>
<br>
{{ prediction }}
</div>
<script
src="{{ url_for('static',filename='main.js') }}"
type="text/javascript"
></script>
</body>
</html>
```

# API Ap

An API app was created for deploying and testing the model

```
app.py index.html APlapp.py X
C: > Users > berly > OneDrive > Desktop > Data Glacier_intern > Week 5 > APlapp.py > ...
1 # Using Flask to make an API
2 # Import the necessary libraries and functions
3 from flask import Flask, jsonify, request
4 import pickle
5 import pandas as pd
6
7 # Creating a Flask app
8 app = Flask(__name__)
9
10 @app.route('/', methods = ['GET', 'POST'])
11 def home ():
12     if(request.method == 'GET'):
13         data = "Hello, this is my first API"
14         return jsonify({'data':data})
15
16
17
18 @app.route('/predict/')
19 def heart_disease_predict():
20     model = pickle.load(open('heart_disease_model.pkl', 'rb'))
21     age = request.args.get('age')
22     sex = request.args.get('sex')
23     cp = request.args.get('cp')
24     trestbps= request.args.get('trestbps')
25     chol = request.args.get('chol')
26     fbs = request.args.get('fbs')
27     restecg = request.args.get('restecg')
28     thalach = request.args.get('thalach')
29     exang = request.args.get('exang')
30     oldpeak = request.args.get('oldpeak')
31     slope = request.args.get('slope')
32     ca = request.args.get('ca')
33     thal = request.args.get('thal')
```

```
34
35 test_df = pd.DataFrame({'Age':[age], 'Sex':[sex], 'Chest Pain Type':[cp], 'Resting Blood Pressure':[trestbps],
36                         'Serum Cholesterol':[chol], 'Fasting Blood Sugar':[fbs], 'Resting electrocardiographic measurement':[restecg],
37                         'Maximum Heart Rate':[thalach], 'Exercise Induce Angina':[exang], 'ST Depression':[oldpeak], 'Slope':[slope],
38                         'Blood Vessel':[ca], 'Thalassemia Blood Disorder':[thal]})
39
40 pred_heart_disease = model.predict(test_df)
41 if pred_heart_disease == 1:
42     return jsonify({'Heart Disease Prediction':"Positive"})
43 else:
44     return jsonify({'Heart Disease Prediction':"Negative"})
45
46 # Driver function
47 if __name__ == '__main__':
48     app.run(debug=True)
```

# API Testing

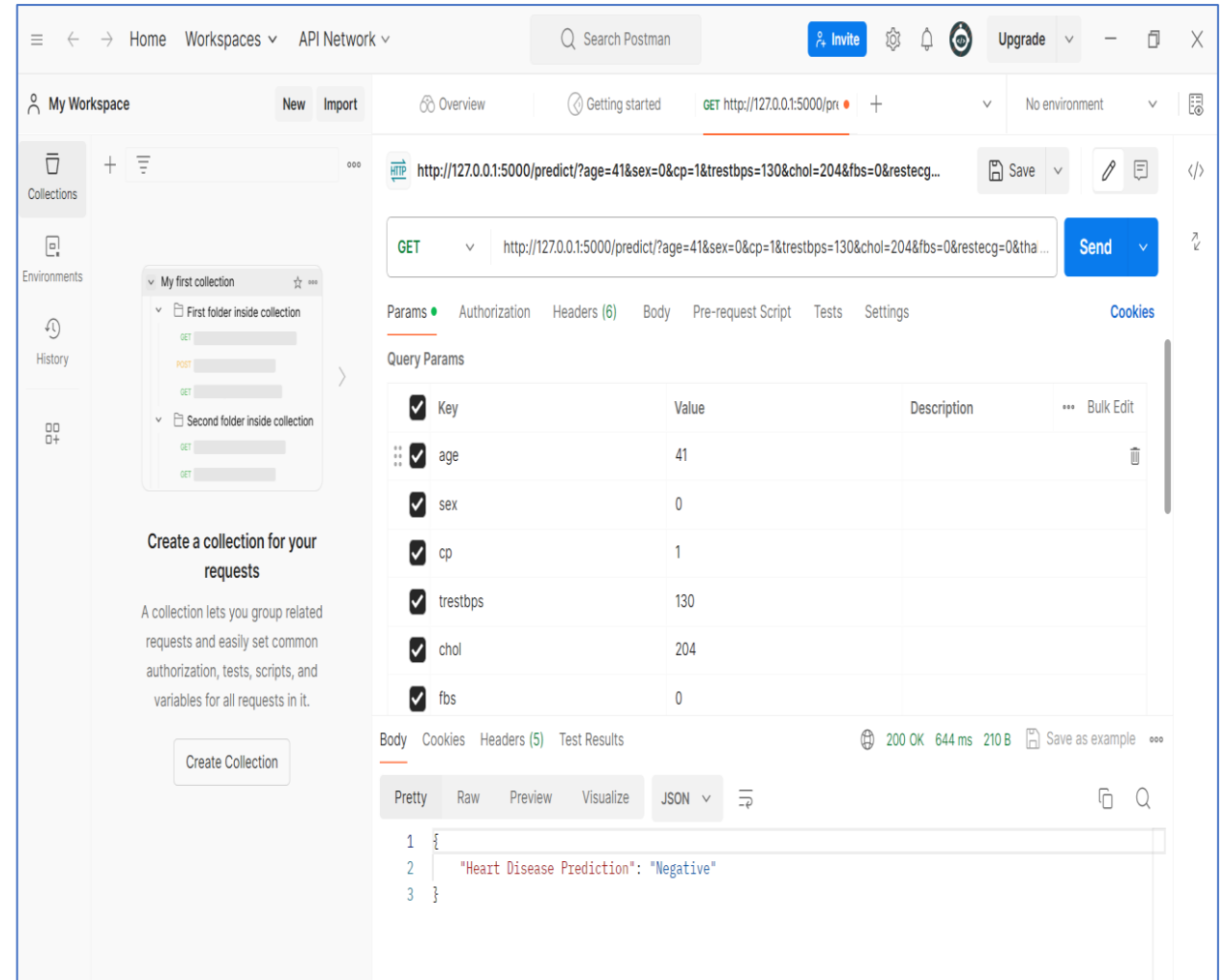
After the creation of all the necessary files, the model was deployed and tested with API

```
Command Prompt - python / X + v
C:\Users\berly\OneDrive>cd desktop
C:\Users\berly\OneDrive\Desktop>cd Data Glacier_intern
C:\Users\berly\OneDrive\Desktop\Data Glacier_intern>cd Week 5
C:\Users\berly\OneDrive\Desktop\Data Glacier_intern\Week 5>dir
Volume in drive C is OS
Volume Serial Number is BAED-4E4F

Directory of C:\Users\berly\OneDrive\Desktop\Data Glacier_intern\Week 5

03/04/2024 15:47 <DIR>      .
02/04/2024 23:45 <DIR>      ..
03/04/2024 15:45          1,827 APIapp.py
29/03/2024 00:17          818 heart_disease_model.pkl
                2 File(s)      2,645 bytes
                2 Dir(s)  284,749,574,144 bytes free

C:\Users\berly\OneDrive\Desktop\Data Glacier_intern\Week 5>python APIapp.py
* Serving Flask app 'APIapp'
* Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on http://127.0.0.1:5000
Press CTRL+C to quit
* Restarting with stat
* Debugger is active!
* Debugger PIN: 274-366-615
```





# Cloud deployment using Heroku

The model was also deployed to the cloud using Heroku by creating the app and connecting to the GitHub repository where all the files have been committed to.

The screenshot shows the Heroku dashboard for the 'heart-disease-predictor' app. The app is connected to the GitHub repository 'Omolar-Alayo/ML-model-deploy'. The 'Deploy' tab is selected, showing options to add the app to a pipeline or deploy manually. The 'Deployment method' section at the bottom shows 'Heroku Git' as the selected method, with 'GitHub' as the provider and 'Connected' status.

Personal > heart-disease-predictor

GitHub Omolar-Alayo/ML-model-deploy

Overview Resources **Deploy** Metrics Activity Access Settings

**Add this app to a pipeline**

Create a new pipeline or choose an existing one and add this app to a stage in it.

**Add this app to a stage in a pipeline to enable additional features**

Pipelines let you connect multiple apps together and **promote code** between them. [Learn more.](#)

Pipelines connected to GitHub can enable **review apps**, and create apps for new pull requests. [Learn more.](#)

Choose a pipeline

**Deployment method**

- Heroku Git Use Heroku CLI
- GitHub Connected**
- Container Registry Use Heroku CLI

The screenshot shows the 'Deploy' settings page for the 'heart-disease-predictor' app. It displays the 'Automatic deploys' section, which is currently disabled. The 'Manual deploy' section is active, showing the 'Deploy a GitHub branch' option. The 'Choose a branch to deploy' dropdown is set to 'main'. The 'Deploy Branch' button is visible. The 'Receive code from GitHub' section shows the 'Build main' status as '8ae9f313'. The 'Release phase' and 'Deploy to Heroku' sections show green checkmarks, indicating successful deployment. A message at the bottom states 'Your app was successfully deployed.' with a 'View' button.

App connected to GitHub

Code diffs, manual and auto deploys are available for this app.

Connected to Omolar-Alayo/ML-model-deploy by Omolar-Alayo [Disconnect...](#)

Releases in the [activity feed](#) link to GitHub to view commit diffs

**Automatic deploys**

Enables a chosen branch to be automatically deployed to this app.

You can now change your main deploy branch from "master" to "main" for both manual and automatic deploys, please follow the instructions [here](#).

Enable automatic deploys from GitHub

Every push to the branch you specify here will deploy a new version of this app. **Deploys happen automatically:** be sure that this branch is always in a deployable state and any tests have passed before you push. [Learn more.](#)

**Choose a branch to deploy**

main

☐ Wait for CI to pass before deploy

Only enable this option if you have a Continuous Integration service configured on your repo.

**Manual deploy**

Deploy the current state of a branch to this app.

**Deploy a GitHub branch**

This will deploy the current state of the branch you specify below. [Learn more.](#)

**Choose a branch to deploy**

main [Deploy Branch](#)

Receive code from GitHub ☒

Build main 8ae9f313 ☒

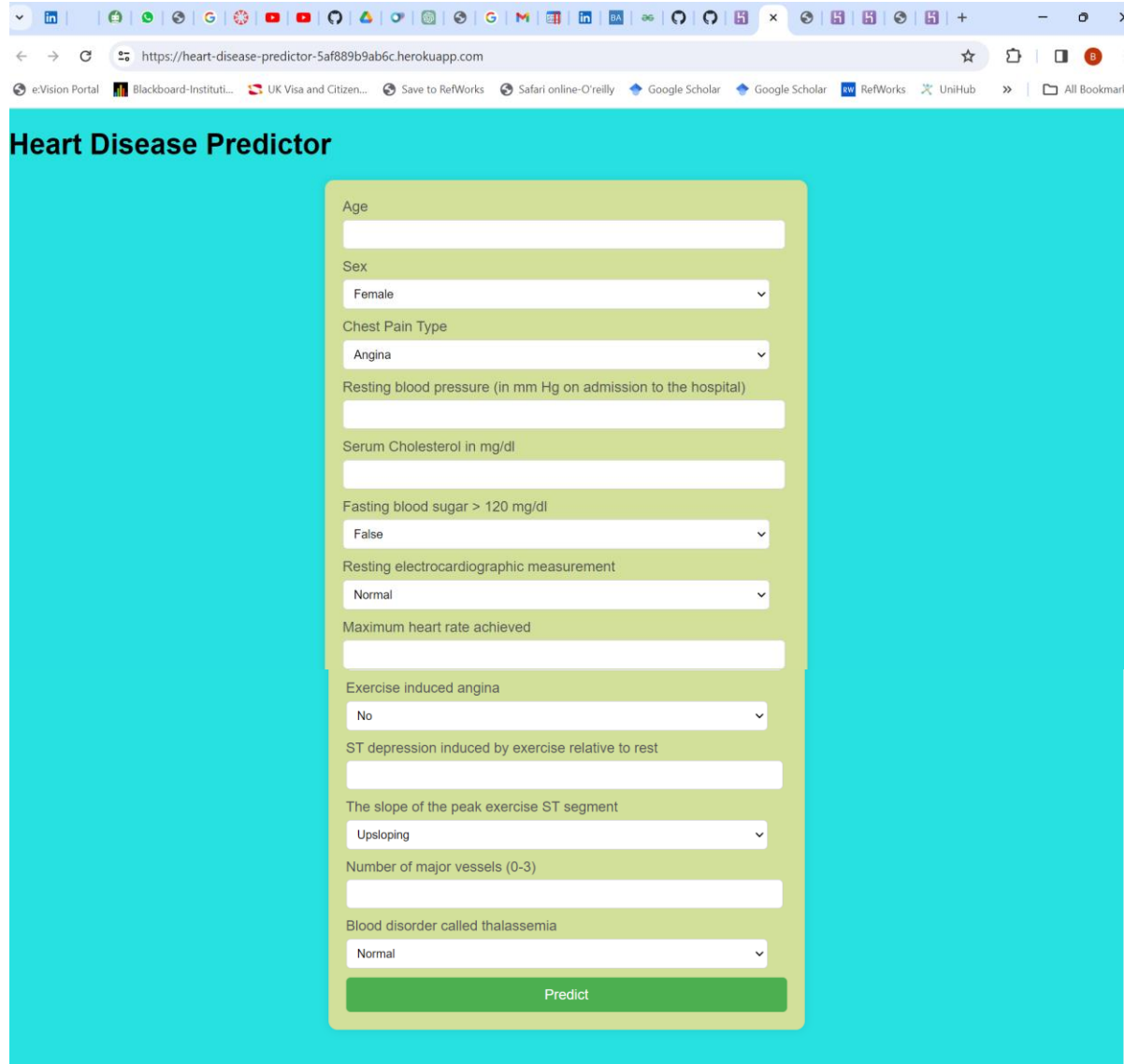
Release phase ☒

Deploy to Heroku ☒

Your app was successfully deployed.

[View](#)

# Heart-disease-predictive app



Heart Disease Predictor

Age

Sex  
Female

Chest Pain Type  
Angina

Resting blood pressure (in mm Hg on admission to the hospital)

Serum Cholesterol in mg/dl

Fasting blood sugar > 120 mg/dl  
False

Resting electrocardiographic measurement  
Normal

Maximum heart rate achieved

Exercise induced angina  
No

ST depression induced by exercise relative to rest

The slope of the peak exercise ST segment  
Upsloping

Number of major vessels (0-3)

Blood disorder called thalassemia  
Normal

Predict

Upon the successful deployment of the model as an app on the web, it was subsequently launched and tested.

# Heart-disease-predictive app

Heart Disease Predictor

Age: 42

Sex: Female

Chest Pain Type: Cardiomyopathy

Resting blood pressure (in mm Hg on admission to the hospital): 130

Serum Cholesterol in mg/dl: 250

Fasting blood sugar > 120 mg/dl: False

Resting electrocardiographic measurement: Normal

Maximum heart rate achieved: 172

Exercise induced angina: No

ST depression induced by exercise relative to rest: 1.4

The slope of the peak exercise ST segment: Upsloping

Number of major vessels (0-3): 0

Blood disorder called thalassemia: Normal

Predict

This person is not suffering from heart disease...

- The web app was hosted on: <https://heart-disease-predictor-5af889b9ab6c.herokuapp.com/>
- It allows any user to enter data and make predictions based on the dataset

Above is the prediction for a particular observation.

# Thank You



**Data Glacier**

Your Deep Learning Partner