Data Intake Report

Name: <Heart Disease Classification>
Report date: <27/3/2022>
Internship Batch:<LISUM07>
Version:<1.0>
Data intake by:<Reema Al-Otaibi>
Data intake reviewer:<intern who reviewed the report>
Data storage location: <GitHub: https://github.com/reemaalotaibi/dataglacier/tree/main/Week3>

Heart.CSV:

Total number of observations	< 918 rows>
Total number of files	<1>
Total number of features	<12 columns>
Base format of the file	<.csv>
Size of the data	<36 KB>

Training the DataSet:

In [120]:
```
features = df[['Age', 'Sex', 'RestingBP','Cholesterol', 'FastingBS', 'MaxHR', 'ExerciseAngina', 'Oldpeak']]
x = df[['Age', 'Sex', 'RestingBP','Cholesterol', 'FastingBS', 'MaxHR', 'ExerciseAngina', 'Oldpeak']]
y = df['HeartDisease']
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size= 0.2, random_state=0)
```

In [121]:
```
model = LogisticRegression()
```

In [122]:
```
model.fit(x_train, y_train)
```

Out[122]:
```
LogisticRegression()
```

In [123]:
```
model.score(x_train, y_train)
```

Out[123]:
```
0.8038147138964578
```

In [124]:
```
predit= model.predict(x_test)
print(predit)
```

```
[1 0 1 1 0 0 0 0 0 0 1 1 1 1 0 1 1 1 1 0 1 1 1 1 0 0 1 1 1 1 1 0 1 0 1 0 1
 0 1 0 0 1 1 0 1 0 0 1 1 0 0 0 0 1 1 1 1 0 1 0 1 0 1 0 1 1 1 0 1 0 0 1 0 1
 1 1 1 0 0 0 1 1 0 0 1 0 0 1 1 1 0 1 0 0 1 0 1 0 0 0 1 0 1 0 1 1 0 0 0 1 1
 1 0 0 1 1 1 1 1 1 1 0 1 0 0 0 0 1 0 1 0 1 1 1 1 1 0 1 1 0 0 1 1 0 1 0 0 1 1
 1 1 0 1 0 0 0 0 1 1 1 0 0 0 1 0 0 1 1 0 0 1 1 1 0 0 0 1 0 1 0 0 1 0 0 1]
```

In [125]:
```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
x_train = scaler.fit_transform(x_train)
x_test = scaler.transform(x_test)
with pd.option_context('display.max_rows', None, 'display.max_columns', None):  # more options can be specified also
    display(x_test)
```

```
array([[ 0.14640812, -0.51740017, -0.95050604, ...,  0.72307486,
        -0.83375663, -0.75922896],
       [ 1.32497163, -0.51740017, -0.67960776, ...,  0.56370636,
        -0.83375663, -0.47271274],
       [-0.28216042, -0.51740017, -1.22140432, ..., -1.74713688,
        -0.83375663, -0.85473437],
       ...,
       [-0.17501829, -0.51740017, -1.11304501, ...,  0.92228548,
        -0.83375663, -0.85473437],
       [-1.78215034, -0.51740017, -0.1378112 , ..., -1.50808413,
        -0.83375663, -0.85473437],
       [-0.06787615, -0.51740017,  1.48757849, ..., -0.59171526,
         1.19939076, -0.85473437]])
```

In [126]:
```
from sklearn import svm
svm = svm.SVC()
svm.fit(x_train, y_train)
predictions = svm.predict(x_test)

from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
print("Confusion Matrix : \n\n" , confusion_matrix(predictions,y_test))
print("Classification Report : \n\n" , classification_report(predictions,y_test),"\n")
```

```
Confusion Matrix :

 [[66 24]
 [11 83]]
Classification Report :

               precision    recall  f1-score   support

           0       0.86      0.73      0.79        90
           1       0.78      0.88      0.83        94

    accuracy                           0.81       184
   macro avg       0.82      0.81      0.81       184
weighted avg       0.82      0.81      0.81       184
```

In [127]:
```
import pickle
pickle.dump(svm, open('model.pkl', 'wb'))
model = pickle.load(open('model.pkl', 'rb'))
print(model)
```

```
SVC()
```

Flask Deployment:

```
import numpy as np
from flask import Flask, request, jsonify, render_template
import pickle
from sklearn.preprocessing import StandardScaler

app = Flask(__name__)

model = pickle.load(open('model.pkl', 'rb'))


@app.route('/')
def home():
    return render_template('index.html')



@app.route('/predict',methods=['POST'])
def predict():

    features = [float(x) for x in request.form.values()]
    final_features = [np.array(features)]
    # final_features = scaler.transform(final_features)
    prediction = model.predict(final_features)


    print("final features",final_features)
    print("prediction:",prediction)
    output = round(prediction[0], 2)
    print(output)

    if output == 0:
        return render_template('index.html', prediction_text='THE PATIENT IS NOT LIKELY TO HAVE A HEART DISEASE')
    else:
         return render_template('index.html', prediction_text='THE PATIENT IS LIKELY TO HAVE A HEART DISEASE')

@app.route('/predict_api',methods=['POST'])
def results():

    data = request.get_json(force=True)
    prediction = model.predict([np.array(list(data.values()))])

    output = prediction[0]
    return jsonify(output)

if __name__ == "__main__":
    app.run(debug=True)

```

HTML Code:

index.html > html > body > div.container.main > div#holder.jumbotron

```
<!DOCTYPE html>
<html lang="en" dir="ltr">
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script>
    <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"></script>
    <link rel="stylesheet" type="text/css" href="static/style.css">
    <title>Heart Failure Prediction</title>
  </head>

  <body>

<div class="container main">
  <div class="jumbotron" id="holder">
     <br>
      <center><b><h2 style="color:#7306bd";>{{prediction_text}}</h2></b></center>
     <br>
     <br>
    <center><h1 class='main_heading'> Heart Disease Classification Model</h1></center>
    <p>This Web Based Application is based on a Machine Learning Algorithm that predicts whether a patient ha
    <br>


    <form class="form-horizontal" action="{{ url_for('predict')}}"method="post">

      <div class="form-group">
        <div class="col-sm-10">
          <input  class="form-control" type= "text" name="Age" placeholder="Age" required="required" />
        </div>
      </div>
```


index.html > html > body > div.container.main > div#holder.jumbotron
33
34
35
36 <input class="form-control" type="text" name="Sex" placeholder="Gender, enter 1 for Female and 0 f
37
38
39
40
41
42 <input class="form-control" type="text" name="ExerciseAngina" placeholder="Exercise Angina, enter
43
44
45
46
47
48
49 <input class="form-control" type="text" name="Cholesterol" placeholder="Cholesterol" required="rec
50
51
52
53
54
55 <input class="form-control" type="text" name="FastingBS" placeholder="FastingBS" required="require
56
57
58
59
60
61 <input class="form-control" type="text" name="MaxHR" placeholder="MaxHR" required="required" />
62
63
64
65
66
67 <input class="form-control" type="text" name="RestingBP" placeholder="RestingBP" required="require
68
69

index.html > html > body > div.container.main > div#holder.jumbotron

```
70
71        <div class="form-group">
72          <div class="col-sm-10">
73            <input  class="form-control" type="text" name="Oldpeak" placeholder="Oldpeak" required="required" /
74          </div>
75        </div>
76
77
78        <div class="form-group">
79          <div class="col-sm-offset-2 col-sm-10">
80           <center><button type="submit" class="button btn btn-default">Discover</button><center/>
81          </div>
82        </div>
83      </form>
84
85      <h2 class="result"></h2>
86    </div>
87    </div>
88
89    </body>
90  </html>
```