**SOURCE CODE**


```
{
 "cells": [
  {
   "cell_type": "markdown",
   "metadata": {},
   "source": [
    "# Importing Required Libraries"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 1,
   "metadata": {},
   "outputs": [],
   "source": [
    "import numpy as np\n",
    "import pandas as pd\n",
    "from sklearn.model_selection import train_test_split\n",
    "from sklearn.linear_model import LogisticRegression\n",
    "from sklearn.metrics import accuracy_score"
   ]
  },
  {
   "cell_type": "markdown",
   "metadata": {},
   "source": [
    "### Data Collection & Processing"
   ]
  },
  {
```

```
 "cell_type": "code",
 "execution_count": 2,
 "metadata": {},
 "outputs": [],
 "source": [
  "data = pd.read_csv(\"heart.csv\")"
 ]
},
{
 "cell_type": "code",
 "execution_count": 3,
 "metadata": {},
 "outputs": [
  {
   "data": {
    "text/html": [
     "<div>\n",
     "<style scoped>\n",
     "    .dataframe tbody tr th:only-of-type {\n",
     "        vertical-align: middle;\n",
     "    }\n",
     "\n",
     "    .dataframe tbody tr th {\n",
     "        vertical-align: top;\n",
     "    }\n",
     "\n",
     "    .dataframe thead th {\n",
     "        text-align: right;\n",
     "    }\n",
     "</style>\n",
     "<table border=\"1\" class=\"dataframe\">\n",
```

```
"  <thead>\n",
"    <tr style=\"text-align: right;\">\n",
"      <th></th>\n",
"      <th>age</th>\n",
"      <th>sex</th>\n",
"      <th>cp</th>\n",
"      <th>trestbps</th>\n",
"      <th>chol</th>\n",
"      <th>fbs</th>\n",
"      <th>restecg</th>\n",
"      <th>thalach</th>\n",
"      <th>exang</th>\n",
"      <th>oldpeak</th>\n",
"      <th>slope</th>\n",
"      <th>ca</th>\n",
"      <th>thal</th>\n",
"      <th>target</th>\n",
"    </tr>\n",
"  </thead>\n",
"  <tbody>\n",
"    <tr>\n",
"      <th>0</th>\n",
"      <td>52</td>\n",
"      <td>1</td>\n",
"      <td>0</td>\n",
"      <td>125</td>\n",
"      <td>212</td>\n",
"      <td>0</td>\n",
"      <td>1</td>\n",
"      <td>168</td>\n",
"      <td>0</td>\n",
```

```
"    <td>1.0</td>\n",
"    <td>2</td>\n",
"    <td>2</td>\n",
"    <td>3</td>\n",
"    <td>0</td>\n",
"   </tr>\n",
"   <tr>\n",
"    <th>1</th>\n",
"    <td>53</td>\n",
"    <td>1</td>\n",
"    <td>0</td>\n",
"    <td>140</td>\n",
"    <td>203</td>\n",
"    <td>1</td>\n",
"    <td>0</td>\n",
"    <td>155</td>\n",
"    <td>1</td>\n",
"    <td>3.1</td>\n",
"    <td>0</td>\n",
"    <td>0</td>\n",
"    <td>3</td>\n",
"    <td>0</td>\n",
"   </tr>\n",
"   <tr>\n",
"    <th>2</th>\n",
"    <td>70</td>\n",
"    <td>1</td>\n",
"    <td>0</td>\n",
"    <td>145</td>\n",
"    <td>174</td>\n",
"    <td>0</td>\n",
```

```
"    <td>1</td>\n",
"    <td>125</td>\n",
"    <td>1</td>\n",
"    <td>2.6</td>\n",
"    <td>0</td>\n",
"    <td>0</td>\n",
"    <td>3</td>\n",
"    <td>0</td>\n",
"   </tr>\n",
"   <tr>\n",
"    <th>3</th>\n",
"    <td>61</td>\n",
"    <td>1</td>\n",
"    <td>0</td>\n",
"    <td>148</td>\n",
"    <td>203</td>\n",
"    <td>0</td>\n",
"    <td>1</td>\n",
"    <td>161</td>\n",
"    <td>0</td>\n",
"    <td>0.0</td>\n",
"    <td>2</td>\n",
"    <td>1</td>\n",
"    <td>3</td>\n",
"    <td>0</td>\n",
"   </tr>\n",
"   <tr>\n",
"    <th>4</th>\n",
"    <td>62</td>\n",
"    <td>0</td>\n",
"    <td>0</td>\n",
```

```
"     <td>138</td>\n",
"     <td>294</td>\n",
"     <td>1</td>\n",
"     <td>1</td>\n",
"     <td>106</td>\n",
"     <td>0</td>\n",
"     <td>1.9</td>\n",
"     <td>1</td>\n",
"     <td>3</td>\n",
"     <td>2</td>\n",
"     <td>0</td>\n",
"    </tr>\n",
"  </tbody>\n",
"</table>\n",
"</div>"
],
"text/plain": [
"   age  sex  cp  trestbps  chol  fbs  restecg  thalach  exang  oldpeak  slope  \\\n",
"0   52    1   0       125   212    0        1      168      0      1.0      2   \n",
"1   53    1   0       140   203    1        0      155      1      3.1      0   \n",
"2   70    1   0       145   174    0        1      125      1      2.6      0   \n",
"3   61    1   0       148   203    0        1      161      0      0.0      2   \n",
"4   62    0   0       138   294    1        1      106      0      1.9      1   \n",
"\n",
"   ca  thal  target  \n",
"0   2     3       0  \n",
"1   0     3       0  \n",
"2   0     3       0  \n",
"3   1     3       0  \n",
"4   3     2       0  "
]
```

```
   },
   "execution_count": 3,
   "metadata": {},
   "output_type": "execute_result"
  }
 ],
 "source": [
  "data.head()"
 ]
},
{
 "cell_type": "code",
 "execution_count": 4,
 "metadata": {},
 "outputs": [
  {
   "data": {
    "text/html": [
     "<div>\n",
     "<style scoped>\n",
     "    .dataframe tbody tr th:only-of-type {\n",
     "        vertical-align: middle;\n",
     "    }\n",
     "\n",
     "    .dataframe tbody tr th {\n",
     "        vertical-align: top;\n",
     "    }\n",
     "\n",
     "    .dataframe thead th {\n",
     "        text-align: right;\n",
     "    }\n",
```

```
"</style>\n",
"<table border=\"1\" class=\"dataframe\">\n",
"  <thead>\n",
"    <tr style=\"text-align: right;\">\n",
"      <th></th>\n",
"      <th>age</th>\n",
"      <th>sex</th>\n",
"      <th>cp</th>\n",
"      <th>trestbps</th>\n",
"      <th>chol</th>\n",
"      <th>fbs</th>\n",
"      <th>restecg</th>\n",
"      <th>thalach</th>\n",
"      <th>exang</th>\n",
"      <th>oldpeak</th>\n",
"      <th>slope</th>\n",
"      <th>ca</th>\n",
"      <th>thal</th>\n",
"      <th>target</th>\n",
"    </tr>\n",
"  </thead>\n",
"  <tbody>\n",
"    <tr>\n",
"      <th>1020</th>\n",
"      <td>59</td>\n",
"      <td>1</td>\n",
"      <td>1</td>\n",
"      <td>140</td>\n",
"      <td>221</td>\n",
"      <td>0</td>\n",
"      <td>1</td>\n",
```

```
"    <td>164</td>\n",
"    <td>1</td>\n",
"    <td>0.0</td>\n",
"    <td>2</td>\n",
"    <td>0</td>\n",
"    <td>2</td>\n",
"    <td>1</td>\n",
"   </tr>\n",
"   <tr>\n",
"    <th>1021</th>\n",
"    <td>60</td>\n",
"    <td>1</td>\n",
"    <td>0</td>\n",
"    <td>125</td>\n",
"    <td>258</td>\n",
"    <td>0</td>\n",
"    <td>0</td>\n",
"    <td>141</td>\n",
"    <td>1</td>\n",
"    <td>2.8</td>\n",
"    <td>1</td>\n",
"    <td>1</td>\n",
"    <td>3</td>\n",
"    <td>0</td>\n",
"   </tr>\n",
"   <tr>\n",
"    <th>1022</th>\n",
"    <td>47</td>\n",
"    <td>1</td>\n",
"    <td>0</td>\n",
"    <td>110</td>\n",
```

```
"    <td>275</td>\n",
"    <td>0</td>\n",
"    <td>0</td>\n",
"    <td>118</td>\n",
"    <td>1</td>\n",
"    <td>1.0</td>\n",
"    <td>1</td>\n",
"    <td>1</td>\n",
"    <td>2</td>\n",
"    <td>0</td>\n",
"   </tr>\n",
"   <tr>\n",
"    <th>1023</th>\n",
"    <td>50</td>\n",
"    <td>0</td>\n",
"    <td>0</td>\n",
"    <td>110</td>\n",
"    <td>254</td>\n",
"    <td>0</td>\n",
"    <td>0</td>\n",
"    <td>159</td>\n",
"    <td>0</td>\n",
"    <td>0.0</td>\n",
"    <td>2</td>\n",
"    <td>0</td>\n",
"    <td>2</td>\n",
"    <td>1</td>\n",
"   </tr>\n",
"   <tr>\n",
"    <th>1024</th>\n",
"    <td>54</td>\n",
```

```
"    <td>1</td>\n",
"    <td>0</td>\n",
"    <td>120</td>\n",
"    <td>188</td>\n",
"    <td>0</td>\n",
"    <td>1</td>\n",
"    <td>113</td>\n",
"    <td>0</td>\n",
"    <td>1.4</td>\n",
"    <td>1</td>\n",
"    <td>1</td>\n",
"    <td>3</td>\n",
"    <td>0</td>\n",
"  </tr>\n",
" </tbody>\n",
"</table>\n",
"</div>"
],
"text/plain": [
"     age  sex  cp  trestbps  chol  fbs  restecg  thalach  exang  oldpeak  \\\n",
"1020   59    1   1       140   221    0        1      164      1      0.0  \n",
"1021   60    1   0       125   258    0        0      141      1      2.8  \n",
"1022   47    1   0       110   275    0        0      118      1      1.0  \n",
"1023   50    0   0       110   254    0        0      159      0      0.0  \n",
"1024   54    1   0       120   188    0        1      113      0      1.4  \n",
"\n",
"     slope  ca  thal  target  \n",
"1020      2   0     2       1  \n",
"1021      1   1     3       0  \n",
"1022      1   1     2       0  \n",
"1023      2   0     2       1  \n",
```

```
    "1024      1   1     3       0  "
   ]
  },
  "execution_count": 4,
  "metadata": {},
  "output_type": "execute_result"
 }
],
"source": [
 "data.tail()"
]
},
{
 "cell_type": "code",
 "execution_count": 5,
 "metadata": {},
 "outputs": [
  {
   "data": {
    "text/plain": [
     "(1025, 14)"
    ]
   },
   "execution_count": 5,
   "metadata": {},
   "output_type": "execute_result"
  }
 ],
 "source": [
  "data.shape"
 ]
```

```
},
{
 "cell_type": "code",
 "execution_count": 6,
 "metadata": {},
 "outputs": [
  {
   "name": "stdout",
   "output_type": "stream",
   "text": [
    "<class 'pandas.core.frame.DataFrame'>\n",
    "RangeIndex: 1025 entries, 0 to 1024\n",
    "Data columns (total 14 columns):\n",
    " #   Column    Non-Null Count  Dtype  \n",
    "---  ------    --------------  -----  \n",
    " 0   age       1025 non-null   int64  \n",
    " 1   sex       1025 non-null   int64  \n",
    " 2   cp        1025 non-null   int64  \n",
    " 3   trestbps  1025 non-null   int64  \n",
    " 4   chol      1025 non-null   int64  \n",
    " 5   fbs       1025 non-null   int64  \n",
    " 6   restecg   1025 non-null   int64  \n",
    " 7   thalach   1025 non-null   int64  \n",
    " 8   exang     1025 non-null   int64  \n",
    " 9   oldpeak   1025 non-null   float64\n",
    " 10  slope     1025 non-null   int64  \n",
    " 11  ca        1025 non-null   int64  \n",
    " 12  thal      1025 non-null   int64  \n",
    " 13  target    1025 non-null   int64  \n",
    "dtypes: float64(1), int64(13)\n",
    "memory usage: 112.2 KB\n"
```

```
   ]
  }
 ],
 "source": [
  "data.info()"
 ]
},
{
 "cell_type": "code",
 "execution_count": 7,
 "metadata": {},
 "outputs": [
  {
   "data": {
    "text/html": [
     "<div>\n",
     "<style scoped>\n",
     "    .dataframe tbody tr th:only-of-type {\n",
     "        vertical-align: middle;\n",
     "    }\n",
     "\n",
     "    .dataframe tbody tr th {\n",
     "        vertical-align: top;\n",
     "    }\n",
     "\n",
     "    .dataframe thead th {\n",
     "        text-align: right;\n",
     "    }\n",
     "</style>\n",
     "<table border=\"1\" class=\"dataframe\">\n",
     "  <thead>\n",
```

```
"    <tr style=\"text-align: right;\">\n",
"      <th></th>\n",
"      <th>age</th>\n",
"      <th>sex</th>\n",
"      <th>cp</th>\n",
"      <th>trestbps</th>\n",
"      <th>chol</th>\n",
"      <th>fbs</th>\n",
"      <th>restecg</th>\n",
"      <th>thalach</th>\n",
"      <th>exang</th>\n",
"      <th>oldpeak</th>\n",
"      <th>slope</th>\n",
"      <th>ca</th>\n",
"      <th>thal</th>\n",
"      <th>target</th>\n",
"    </tr>\n",
"  </thead>\n",
"  <tbody>\n",
"    <tr>\n",
"      <th>count</th>\n",
"      <td>1025.000000</td>\n",
"      <td>1025.000000</td>\n",
"      <td>1025.000000</td>\n",
"      <td>1025.000000</td>\n",
"      <td>1025.00000</td>\n",
"      <td>1025.000000</td>\n",
"      <td>1025.000000</td>\n",
"      <td>1025.000000</td>\n",
"      <td>1025.000000</td>\n",
"      <td>1025.000000</td>\n",
```

```
"    <td>1025.000000</td>\n",
"    <td>1025.000000</td>\n",
"    <td>1025.000000</td>\n",
"    <td>1025.000000</td>\n",
"   </tr>\n",
"   <tr>\n",
"    <th>mean</th>\n",
"    <td>54.434146</td>\n",
"    <td>0.695610</td>\n",
"    <td>0.942439</td>\n",
"    <td>131.611707</td>\n",
"    <td>246.00000</td>\n",
"    <td>0.149268</td>\n",
"    <td>0.529756</td>\n",
"    <td>149.114146</td>\n",
"    <td>0.336585</td>\n",
"    <td>1.071512</td>\n",
"    <td>1.385366</td>\n",
"    <td>0.754146</td>\n",
"    <td>2.323902</td>\n",
"    <td>0.513171</td>\n",
"   </tr>\n",
"   <tr>\n",
"    <th>std</th>\n",
"    <td>9.072290</td>\n",
"    <td>0.460373</td>\n",
"    <td>1.029641</td>\n",
"    <td>17.516718</td>\n",
"    <td>51.59251</td>\n",
"    <td>0.356527</td>\n",
"    <td>0.527878</td>\n",
```

```
"    <td>23.005724</td>\n",
"    <td>0.472772</td>\n",
"    <td>1.175053</td>\n",
"    <td>0.617755</td>\n",
"    <td>1.030798</td>\n",
"    <td>0.620660</td>\n",
"    <td>0.500070</td>\n",
"   </tr>\n",
"   <tr>\n",
"    <th>min</th>\n",
"    <td>29.000000</td>\n",
"    <td>0.000000</td>\n",
"    <td>0.000000</td>\n",
"    <td>94.000000</td>\n",
"    <td>126.00000</td>\n",
"    <td>0.000000</td>\n",
"    <td>0.000000</td>\n",
"    <td>71.000000</td>\n",
"    <td>0.000000</td>\n",
"    <td>0.000000</td>\n",
"    <td>0.000000</td>\n",
"    <td>0.000000</td>\n",
"    <td>0.000000</td>\n",
"    <td>0.000000</td>\n",
"   </tr>\n",
"   <tr>\n",
"    <th>25%</th>\n",
"    <td>48.000000</td>\n",
"    <td>0.000000</td>\n",
"    <td>0.000000</td>\n",
"    <td>120.000000</td>\n",
```

```
"    <td>211.00000</td>\n",
"    <td>0.000000</td>\n",
"    <td>0.000000</td>\n",
"    <td>132.000000</td>\n",
"    <td>0.000000</td>\n",
"    <td>0.000000</td>\n",
"    <td>1.000000</td>\n",
"    <td>0.000000</td>\n",
"    <td>2.000000</td>\n",
"    <td>0.000000</td>\n",
"   </tr>\n",
"   <tr>\n",
"    <th>50%</th>\n",
"    <td>56.000000</td>\n",
"    <td>1.000000</td>\n",
"    <td>1.000000</td>\n",
"    <td>130.000000</td>\n",
"    <td>240.00000</td>\n",
"    <td>0.000000</td>\n",
"    <td>1.000000</td>\n",
"    <td>152.000000</td>\n",
"    <td>0.000000</td>\n",
"    <td>0.800000</td>\n",
"    <td>1.000000</td>\n",
"    <td>0.000000</td>\n",
"    <td>2.000000</td>\n",
"    <td>1.000000</td>\n",
"   </tr>\n",
"   <tr>\n",
"    <th>75%</th>\n",
"    <td>61.000000</td>\n",
```

```
"    <td>1.000000</td>\n",
"    <td>2.000000</td>\n",
"    <td>140.000000</td>\n",
"    <td>275.00000</td>\n",
"    <td>0.000000</td>\n",
"    <td>1.000000</td>\n",
"    <td>166.000000</td>\n",
"    <td>1.000000</td>\n",
"    <td>1.800000</td>\n",
"    <td>2.000000</td>\n",
"    <td>1.000000</td>\n",
"    <td>3.000000</td>\n",
"    <td>1.000000</td>\n",
"   </tr>\n",
"   <tr>\n",
"    <th>max</th>\n",
"    <td>77.000000</td>\n",
"    <td>1.000000</td>\n",
"    <td>3.000000</td>\n",
"    <td>200.000000</td>\n",
"    <td>564.00000</td>\n",
"    <td>1.000000</td>\n",
"    <td>2.000000</td>\n",
"    <td>202.000000</td>\n",
"    <td>1.000000</td>\n",
"    <td>6.200000</td>\n",
"    <td>2.000000</td>\n",
"    <td>4.000000</td>\n",
"    <td>3.000000</td>\n",
"    <td>1.000000</td>\n",
"   </tr>\n",
```

```
"  </tbody>\n",
"</table>\n",
"</div>"
],
"text/plain": [
"            age         sex          cp    trestbps        chol  \\\n",
"count  1025.000000  1025.000000  1025.000000  1025.000000  1025.00000  \n",
"mean     54.434146     0.695610     0.942439   131.611707   246.00000  \n",
"std       9.072290     0.460373     1.029641    17.516718    51.59251  \n",
"min      29.000000     0.000000     0.000000    94.000000   126.00000  \n",
"25%      48.000000     0.000000     0.000000   120.000000   211.00000  \n",
"50%      56.000000     1.000000     1.000000   130.000000   240.00000  \n",
"75%      61.000000     1.000000     2.000000   140.000000   275.00000  \n",
"max      77.000000     1.000000     3.000000   200.000000   564.00000  \n",
"\n",
"             fbs      restecg      thalach        exang      oldpeak  \\\n",
"count  1025.000000  1025.000000  1025.000000  1025.000000  1025.000000  \n",
"mean      0.149268     0.529756   149.114146     0.336585     1.071512  \n",
"std       0.356527     0.527878    23.005724     0.472772     1.175053  \n",
"min       0.000000     0.000000    71.000000     0.000000     0.000000  \n",
"25%       0.000000     0.000000   132.000000     0.000000     0.000000  \n",
"50%       0.000000     1.000000   152.000000     0.000000     0.800000  \n",
"75%       0.000000     1.000000   166.000000     1.000000     1.800000  \n",
"max       1.000000     2.000000   202.000000     1.000000     6.200000  \n",
"\n",
"            slope         ca       thal      target  \n",
"count  1025.000000  1025.000000  1025.000000  1025.000000  \n",
"mean      1.385366     0.754146     2.323902     0.513171  \n",
"std       0.617755     1.030798     0.620660     0.500070  \n",
"min       0.000000     0.000000     0.000000     0.000000  \n",
"25%       1.000000     0.000000     2.000000     0.000000  \n",
```

```
    "50%      1.000000    0.000000    2.000000    1.000000  \n",
    "75%      2.000000    1.000000    3.000000    1.000000  \n",
    "max      2.000000    4.000000    3.000000    1.000000  "
   ]
  },
  "execution_count": 7,
  "metadata": {},
  "output_type": "execute_result"
 }
],
"source": [
 "data.describe()"
]
},
{
 "cell_type": "code",
 "execution_count": 8,
 "metadata": {},
 "outputs": [
  {
   "data": {
    "text/plain": [
     "1    526\n",
     "0    499\n",
     "Name: target, dtype: int64"
    ]
   },
   "execution_count": 8,
   "metadata": {},
   "output_type": "execute_result"
  }
```

```
 ],
 "source": [
  "# checking the distribution of Target Variable\n",
  "data['target'].value_counts()"
 ]
},
{
 "cell_type": "markdown",
 "metadata": {},
 "source": [
  "### 1 --> Defective Heart\n",
  "\n",
  "### 0 --> Healthy Heart"
 ]
},
{
 "cell_type": "markdown",
 "metadata": {},
 "source": [
  "# Data Splitting(Features & Target)"
 ]
},
{
 "cell_type": "code",
 "execution_count": 9,
 "metadata": {},
 "outputs": [],
 "source": [
  "X = data.drop(columns='target', axis=1)\n",
  "Y = data['target']"
 ]
```

```
},
{
 "cell_type": "code",
 "execution_count": 10,
 "metadata": {},
 "outputs": [
  {
   "name": "stdout",
   "output_type": "stream",
   "text": [
    "     age  sex  cp  trestbps  chol  fbs  restecg  thalach  exang  oldpeak  \\\n",
    "0     52    1   0     125   212    0      1     168     0     1.0   \n",
    "1     53    1   0     140   203    1      0     155     1     3.1   \n",
    "2     70    1   0     145   174    0      1     125     1     2.6   \n",
    "3     61    1   0     148   203    0      1     161     0     0.0   \n",
    "4     62    0   0     138   294    1      1     106     0     1.9   \n",
    "...   ...  ...  ..     ...   ...  ...    ...     ...   ...     ...   \n",
    "1020  59    1   1     140   221    0      1     164     1     0.0   \n",
    "1021  60    1   0     125   258    0      0     141     1     2.8   \n",
    "1022  47    1   0     110   275    0      0     118     1     1.0   \n",
    "1023  50    0   0     110   254    0      0     159     0     0.0   \n",
    "1024  54    1   0     120   188    0      1     113     0     1.4   \n",
    "\n",
    "     slope  ca  thal  \n",
    "0        2   2     3  \n",
    "1        0   0     3  \n",
    "2        0   0     3  \n",
    "3        2   1     3  \n",
    "4        1   3     2  \n",
    "...    ...  ..   ...  \n",
    "1020     2   0     2  \n",
```

```
   "1021     1   1     3  \n",
   "1022     1   1     2  \n",
   "1023     2   0     2  \n",
   "1024     1   1     3  \n",
   "\n",
   "[1025 rows x 13 columns]\n"
  ]
 }
],
"source": [
 "print(X)"
]
},
{
"cell_type": "code",
"execution_count": 11,
"metadata": {},
"outputs": [
 {
  "name": "stdout",
  "output_type": "stream",
  "text": [
   "0      0\n",
   "1      0\n",
   "2      0\n",
   "3      0\n",
   "4      0\n",
   "      ..\n",
   "1020    1\n",
   "1021    0\n",
   "1022    0\n",
```

```
   "1023    1\n",
   "1024    0\n",
   "Name: target, Length: 1025, dtype: int64\n"
  ]
 }
],
"source": [
 "print(Y)"
]
},
{
 "cell_type": "markdown",
 "metadata": {},
 "source": [
  "### Splitting the Data into Training data & Test Data"
 ]
},
{
 "cell_type": "code",
 "execution_count": 12,
 "metadata": {},
 "outputs": [],
 "source": [
  "X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, stratify=Y, random_state=2)"
 ]
},
{
 "cell_type": "code",
 "execution_count": 13,
 "metadata": {},
 "outputs": [
```

```
 {
  "name": "stdout",
  "output_type": "stream",
  "text": [
   "(1025, 13) (820, 13) (205, 13)\n"
  ]
 }
],
"source": [
 "print(X.shape, X_train.shape, X_test.shape)"
]
},
{
"cell_type": "markdown",
"metadata": {},
"source": [
 "# Model Training"
]
},
{
"cell_type": "markdown",
"metadata": {},
"source": [
 "### Logistic Regression"
]
},
{
"cell_type": "code",
"execution_count": 14,
"metadata": {},
"outputs": [],
```

```
 "source": [
  "model = LogisticRegression()"
 ]
},
{
 "cell_type": "code",
 "execution_count": 15,
 "metadata": {},
 "outputs": [
  {
   "name": "stderr",
   "output_type": "stream",
   "text": [
    "C:\\Users\\hp\\AppData\\Local\\Programs\\Python\\Python310\\lib\\site-packages\\sklearn\\linear_model\\_logistic.py:814: ConvergenceWarning: lbfgs failed to converge (status=1):\n",
    "STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.\n",
    "\n",
    "Increase the number of iterations (max_iter) or scale the data as shown in:\n",
    "    https://scikit-learn.org/stable/modules/preprocessing.html\n",
    "Please also refer to the documentation for alternative solver options:\n",
    "    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression\n",
    "  n_iter_i = _check_optimize_result(\n"
   ]
  },
  {
   "data": {
    "text/plain": [
     "LogisticRegression()"
    ]
   },
   "execution_count": 15,
```

```
   "metadata": {},
   "output_type": "execute_result"
  }
 ],
 "source": [
  "model.fit(X_train, Y_train)"
 ]
},
{
 "cell_type": "markdown",
 "metadata": {},
 "source": [
  "## Evaluation"
 ]
},
{
 "cell_type": "markdown",
 "metadata": {},
 "source": [
  "### Accuracy Score"
 ]
},
{
 "cell_type": "code",
 "execution_count": 16,
 "metadata": {},
 "outputs": [],
 "source": [
  "X_train_prediction= model.predict(X_train)\n",
  "training_data_accuracy = accuracy_score(X_train_prediction, Y_train)"
 ]
```

```
},
{
 "cell_type": "code",
 "execution_count": 17,
 "metadata": {},
 "outputs": [
  {
   "name": "stdout",
   "output_type": "stream",
   "text": [
    "Accuracy on Training data :  0.8524390243902439\n"
   ]
  }
 ],
 "source": [
  "print('Accuracy on Training data : ', training_data_accuracy)"
 ]
},
{
 "cell_type": "code",
 "execution_count": 18,
 "metadata": {},
 "outputs": [],
 "source": [
  "X_test_prediction = model.predict(X_test)\n",
  "test_data_accuracy = accuracy_score(X_test_prediction, Y_test)"
 ]
},
{
 "cell_type": "code",
 "execution_count": 19,
```

```
 "metadata": {},
 "outputs": [
  {
   "name": "stdout",
   "output_type": "stream",
   "text": [
    "Accuracy on Test data :  0.8048780487804879\n"
   ]
  }
 ],
 "source": [
  "print('Accuracy on Test data : ', test_data_accuracy)"
 ]
},
{
 "cell_type": "markdown",
 "metadata": {},
 "source": [
  "## Predictive System"
 ]
},
{
 "cell_type": "code",
 "execution_count": 21,
 "metadata": {},
 "outputs": [
  {
   "name": "stdout",
   "output_type": "stream",
   "text": [
    "[0]\n",
```

```
    "The Person does not have a Heart Disease\n"
   ]
  },
  {
   "name": "stderr",
   "output_type": "stream",
   "text": [
    "C:\\Users\\hp\\AppData\\Local\\Programs\\Python\\Python310\\lib\\site-packages\\sklearn\\base.py:450: UserWarning: X does not have valid feature names, but LogisticRegression was fitted with feature names\n",
    "  warnings.warn(\n"
   ]
  }
 ],
 "source": [
  "input_data = (3\t,61\t,1\t,0\t,148\t,203,\t0\t,1\t,161,0,\t2,\t1,\t3\t\t)\n",
  "\n",
  "input_data_as_numpy_array= np.asarray(input_data)\n",
  "\n",
  "\n",
  "input_data_reshaped = input_data_as_numpy_array.reshape(1,-1)\n",
  "\n",
  "prediction = model.predict(input_data_reshaped)\n",
  "print(prediction)\n",
  "\n",
  "if (prediction[0]== 0):\n",
  "  print('The Person does not have a Heart Disease')\n",
  "else:\n",
  "  print('The Person has Heart Disease')"
 ]
 }
],
```

```
 "metadata": {
  "interpreter": {
   "hash": "a25b2d42e0fe57251b6ece2b75d30429fe26895efea1faac6949f166a7477be7"
  },
  "kernelspec": {
   "display_name": "Python 3 (ipykernel)",
   "language": "python",
   "name": "python3"
  },
  "language_info": {
   "codemirror_mode": {
    "name": "ipython",
    "version": 3
   },
   "file_extension": ".py",
   "mimetype": "text/x-python",
   "name": "python",
   "nbconvert_exporter": "python",
   "pygments_lexer": "ipython3",
   "version": "3.9.7"
  }
 },
 "nbformat": 4,
 "nbformat_minor": 2
}
```