Heart Disease Prediction System using Machine Learning

Ankit Nainwal · Follow
6 min read · Jan 7

A Heart Disease Prediction System built on machine learning

Photo by Matt Collamer on Unsplash

Principle

The heart disease prediction system uses a classification model to predict whether a person is likely to have heart disease. The model is trained on a dataset of patient records that includes information such as **age, gender, cholesterol levels, blood pressure, and number of major vessesls**. The model uses this information to learn the relationship between these factors and the likelihood of heart disease.

Once the model is trained, it can be used to predict the likelihood of heart disease for new patients. The model does this by taking the patient's information and using it to calculate a probability. If the probability is high, the model predicts that the patient is likely to have heart disease. If the probability is low, the model predicts that the patient is not likely to have heart disease.

The principle of working of the heart disease prediction system is as follows:

1. The system collects data from patients, such as their age, gender, cholesterol levels, blood pressure, and smoking status.
2. The system uses this data to train a 5 classification models.
3. The model learns the relationship between the patient's data and the likelihood of heart disease.
4. One of the models (with good accuracy) is used to predict the likelihood of heart disease for new patients.

The system is still under development, but it has the potential to be a valuable tool for early detection of heart disease. By identifying patients who are at risk, the system can help to prevent heart disease and improve patient outcomes.

Short Description of project

Here is a short description of the https://github.com/nano-bot01/Heart-Disease-Prediction-System-using-Machine-Learning project:

DEPLOYED_HERE

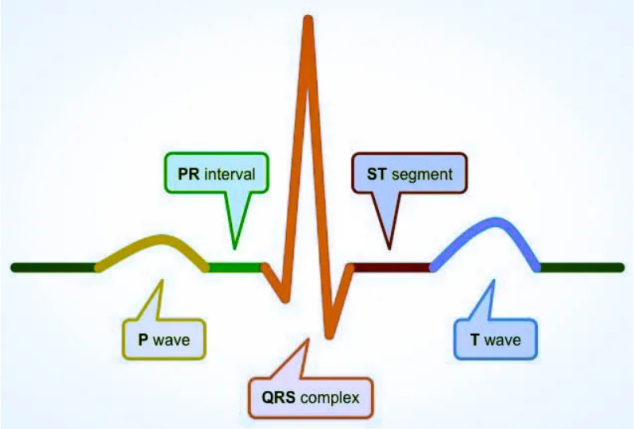
This project is a heart disease prediction system that uses 5classifiction models to predict whether a person has heart disease or not. The system is built using Python and Flask, and it is deployed on a local machine. The system uses the Heart Disease dataset, which is a publicly available dataset of heart disease patient records. The system has an accuracy of 87%, which is comparable to the accuracy of other heart disease prediction systems.

The project is well-documented, and it includes a README file that provides instructions on how to run the system. The project also includes a Jupyter notebook that can be used to train and evaluate the logistic regression model.

Here are some of the features of the project:

- Uses a logistic regression model to predict heart disease
- Built using Python and Flask
- Deployed on a cloud
- Uses the Heart Disease UCI dataset
- Has an accuracy of 87% with k-nearest neighbor
- Well-documented
- Includes a README file and a Jupyter notebook

What is ECG ??



An electrocardiogram (ECG) is a quick test that can be used to examine the electrical activity and rhythm of your heart.
The electrical signals that your heart beats out each time it beats are picked up by sensors that are affixed to your skin.
A machine records these signals, and a doctor examines them to see whether they are odd.

Workflow of model

- Data collection
- Data Visualization
- Splitting the Features and Target
- Train-Test split

- Model Training
- Model Evaluation
- Predicting Results
- Saving Model

Data collection

Dataset link

Dataset information

1. age (in numbers)
2. sex (0 : female, 1 : male)
3. chest pain type (4 values : 0–3)
4. Resting blood pressure (numeric only)
5. Serum Cholestoral in mg/dl
6. Fasting blood sugar > 120 mg/dl
7. Resting electrocardiographic results (values 0,1,2)
8. Maximum heart rate achieved
9. Exercise induced angina
10. Oldpeak = ST depression induced by exercise relative to rest
11. The slope of the peak exercise ST segment
12. Number of major vessels (0–3) colored by flourosopy
13. Thal: 0 = normal; 1 = fixed defect; 2 = reversable defect

Output : Either Heart Disease is present of not (0 or 1)

Dependencies

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

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import GaussianNB
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC

from sklearn import metrics
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_sc

import joblib
```

Split Features and Target set

```
# loading data into pandas data frame

heart_data = pd.read_csv("/content/heart.csv")
heart_data.head(10)

# columns name

heart_data.columns

# checking for missing values

heart_data.isnull().sum()
```

Splitting the Features and Target

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

```
X = heart_data.drop(columns = 'target', axis = 1)
X.head()

# now X contains table without target column which will help for training the da
```

Train Test Split

```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.15, stra

# stratify will distribute 0 and 1 in even manner, of that prediction will be un
# test_split tells a ratio about size of test data in dataset, means 15 percent
# random_state tells about the randomness of data, and number tells about its ex
```

Model Training

```
# instantiate the model
lr = LogisticRegression()

# training the LogisticRegression model with training data
lr.fit(X_train, Y_train)

y_pred = lr.predict(X_test)

print('Model accuracy score: {0:0.4f}'. format(accuracy_score(Y_test, y_pred)))


# instantiate the model
gnb = GaussianNB()
model = gnb

# fit the model
gnb.fit(X_train, Y_train)
y_pred = gnb.predict(X_test)

y_pred
print('Model accuracy score: {0:0.4f}'. format(accuracy_score(Y_test, y_pred)))


# instantiate the model
knn = KNeighborsClassifier(n_neighbors=7)


# fit the model
knn.fit(X_train, Y_train)
y_pred = knn.predict(X_test)

y_pred
print('Model accuracy score: {0:0.4f}'. format(accuracy_score(Y_test, y_pred)))
```

Model Evaluation

1. Accuracy Score

```
# accuracy of traning data
# accuracy function measures accuracy between two values,or columns

X_train_prediction = knn.predict(X_train)
training_data_accuracy = accuracy_score(X_train_prediction, Y_train)

print("The accuracy of training data : ", training_data_accuracy)

# accuracy of traning data
# accuracy function measures accuracy between two values,or columns

X_train_prediction = knn.predict(X_train)
training_data_accuracy = accuracy_score(X_train_prediction, Y_train)

print("The accuracy of training data : ", training_data_accuracy)
```

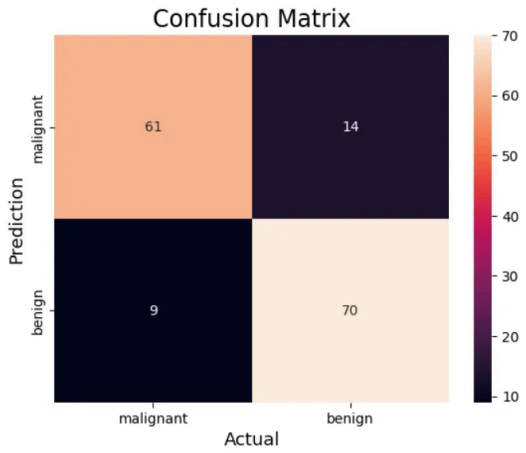
2. Metrics

```
# Accuracy, F1, Recall, Precision

Y_pred = knn.predict(X_test)
```

```
accuracy = accuracy_score(Y_test, Y_pred)
print("Accuracy   :", accuracy)
precision = precision_score(Y_test, Y_pred)
print("Precision  :", precision)
recall = recall_score(Y_test, Y_pred)
print("Recall     :", recall)
F1_score = f1_score(Y_test, Y_pred)
print("F1-score   :", F1_score)

# check results
print(metrics.classification_report(Y_test, Y_pred))
```

3. Confusion Metrix

```
# confusion matrix

cm = confusion_matrix(Y_test,Y_pred)

#Plot the confusion matrix.
sns.heatmap(cm,
            annot=True,
            fmt='g',
            xticklabels=['malignant', 'benign'],
            yticklabels=['malignant', 'benign'])
plt.ylabel('Prediction',fontsize=13)
plt.xlabel('Actual',fontsize=13)
plt.title('Confusion Matrix',fontsize=17)
plt.show()
```



Predicting Results

Steps :

— Take input data
— Process the data, change into array
— Reshape data as single element in array
— Predict output using predict function
— Output the value

```
# input feature values
input_data = (42,1,0,136,315,0,1,125,1,1.8,1,0,1)
# change the input data into a numpy array
input_data_as_numpy_array = np.array(input_data)
# reshape the array to predict data for only one instance
reshaped_array = input_data_as_numpy_array.reshape(1,-1)
```

Printing Results

```
# predicting the result and printing it
prediction = model.predict(reshaped_array)
print(prediction)
if(prediction[0] == 0):
```

```
 print("The Patient has a healthy heart 💛💛💛💛")
else:
 print("The Patient has an unhealthy heart 💔💔💔💔")
```

Notations of predicted output:

— [0] : means patient has a healthy heart 💛 💛 💛 💛
— [1] : means patient has a unhealthy heart 💔 💔 💔 💔

Saving the model

```
import pickle
# importing the library

filename = "trained_model.pkl"
pickle.dump(model, open(filename, 'wb'))
# saving file
```

Repository Link

Deployed Link

Contributor:

Ankit Nainwal

Twitter || LinkedIn || Hashnode || Instagram

Please Like and ⭐ on github.

Follow for more. ⭐ ⭐ .

Machine Learning | Regression | Prediction System | Supervised Learning | Open Source

52

Written by Ankit Nainwal

Follow

11 Followers

I write about Machine Learning, Deep Learning and awesome web technologies. I am an undergraduate learner finding his passion in this dyanamic world.

More from Ankit Nainwal

Ankit Nainwal

Neural Networks

Article explaining each and everything about Neural Network

8 min read · Mar 18

Ankit Nainwal

K Means Clustering on IRIS Dataset

K- Means Clustering

3 min read · Feb 9

54 25

Ankit Nainwal

Java Loops Concepts!

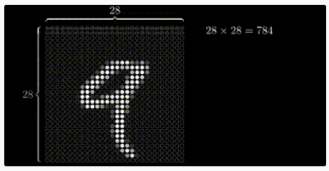
Loops origin, basic concepts with 6 problems based on loops.

5 min read · Feb 2

33

See all from Ankit Nainwal

Recommended from Medium



Sadaf Saleem

Neural Networks in 10mins. Simply Explained!

What are Neural Networks?

9 min read · May 15

549 4

Eryk Lewinson in Towards Data Science

The Comprehensive Guide to Moving Averages in Time Series...

Exploring the Nuances of Simple Moving Averages and Exponentially Weighted...

· 9 min read · 6 days ago

146 1

Lists

Predictive Modeling w/ Python

20 stories · 562 saves

Practical Guides to Machine Learning

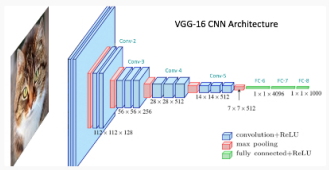
10 stories · 641 saves

Natural Language Processing

799 stories · 367 saves

The New Chatbots: ChatGPT, Bard, and Beyond

12 stories · 180 saves

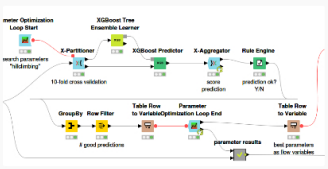


Luís Fernando Torres in LatinXinAI

Convolutional Neural Network From Scratch

The most effective way of working with image data

21 min read · Oct 16



Hans Samson in Low Code for Data Science

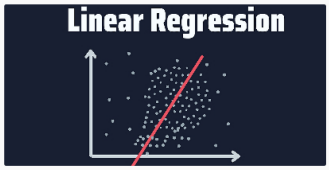
Can I predict the outcome of a football match (and make money)?

Experiments in sport analytics using KNIME Analytics Platform

11 min read · Jun 26

373 4

209 1



Yennhi95zz

#2. Introduction to Linear Regression: Predicting House...

In this article, we'll explore the concept of linear regression, one of the fundamental...

4 min read · May 24

64 1

Ryuta Yoshimatsu

Causal ML for Decision Making

Introduction and Application

15 min read · Oct 25

149

See more recommendations