HEART DISEASE PREDICTION SYSTEM

Viraj Parab

Date: 02-10-2022

1. Problem Statement

The major challenge in heart disease is its detection. There are instruments available which can predict heart disease but either they are expensive or are not efficient to calculate chance of heart disease in human. Early detection of cardiac diseases can decrease the mortality rate and overall complications. However, it is not possible to monitor patients every day in all cases accurately and consultation of a patient for 24 hours by a doctor is not available since it requires more sapience, time and expertise.

2. Market/Customer Need Assessment

India has one of the highest burdens of cardiovascular disease (CVD) worldwide. The annual number of deaths from CVD in India is projected to rise from 2.26 million (1990) to 4.77 million (2020). Coronary heart disease prevalence rates in India have been estimated over the past several decades and have ranged from 1.6% to 7.4% in rural populations and from 1% to 13.2% in urban populations.
Our aim is to predict the presence of heart disease in the patient with the help of Machine Learning Algorithms. This will help individuals as well as the doctors to get the early idea about it which will help them to take the precautions accordingly.
This will help reduce the risk of heart attack, decrease the mortality rate and overall complications.

3. Target Specification and characterization

A. To change traditional heart disease prediction process to faster and accurate process.
B. Reducing frustration and death of patients due to delay in the prediction.
C. Predetermined dataset of heart disease patients and normal patients is taken and based on that prediction is performed.
Above, mentioned targets can be achieved by analyzing:
1. What the patient looks for
2. How are present heart disease prediction processes are being performed
3. Problems faced by people suffering from heart disease
4. How to identify and provide treatment in initial stage accurately.
5. How efficiently are the cardiac surgeons performing heart surgery.
6. When and where a patient likes to trust and spend on.
7. Analyzing the needs of the patients suffering from heart disease.
8. To help patient fight heart disease in early stage
9. To send results to the patient within minutes and prescribing the next step to be taken by the patient.
(if he’s been found of suffering from heart disease)
10. To remind the patient about the latest changes in the heart operations.

4. External Search (Information sources)

The dataset can be found on the Kaggle
(link : https://www.kaggle.com/datasets/johnsmith88/heart-disease-dataset)
This data set dates from 1988 and consists of four databases: Cleveland, Hungary, Switzerland, and Long Beach V. It contains 76 attributes, including the predicted attribute, but all published experiments refer to using a subset of 14 of them. The "target" field refers to the presence of heart disease in the patient. It is integer valued 0 = no disease and 1 = disease.

Importing Libraries

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

import warnings
warnings.filterwarnings("ignore")
```

Reading the data

In [2]:
```
df = pd.read_csv("C:/Users/Viraj/OneDrive/Documents/Birla 2nd Sem/ML/Files/heart.csv")
```

In [3]:
```
df.head() #top 5 rows
```

Out[3]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	52	1	0	125	212	0	1	168	0	1.0	2	2	3	0
1	53	1	0	140	203	1	0	155	1	3.1	0	0	3	0
2	70	1	0	145	174	0	1	125	1	2.6	0	0	3	0
3	61	1	0	148	203	0	1	161	0	0.0	2	1	3	0
4	62	0	0	138	294	1	1	106	0	1.9	1	3	2	0

Attribute Information:

```
1.age
2.sex (1= Male, 0= Female)
3.chest pain type (4 values)
4.resting blood pressure
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
```

In [4]:
```
df.shape
```

Out[4]:
```
(1025, 14)
```

There are total 1025 rows and 14 columns

In [5]:
```
df.info()
```

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

If we see the datatypes of the attributes, we can notice that all datatypes are integer datatypes except the one of oldpeak attribute which is float datatype.

5. Benchmarking

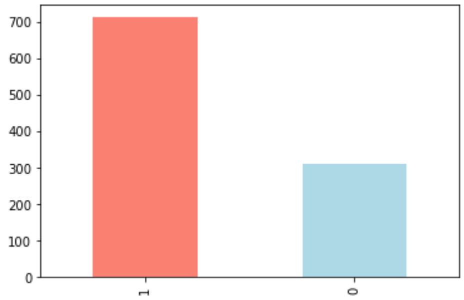
EDA

In [53]:
```
df['sex'].value_counts()
```

Out[53]:
```
1    713
0    312
Name: sex, dtype: int64
```

In [52]:
```
df['sex'].value_counts().plot(kind='bar', color=['salmon', 'lightblue'])
```

Out[52]: <AxesSubplot:>



Out of 1025 records, 713 records are of males and 312 records are of females

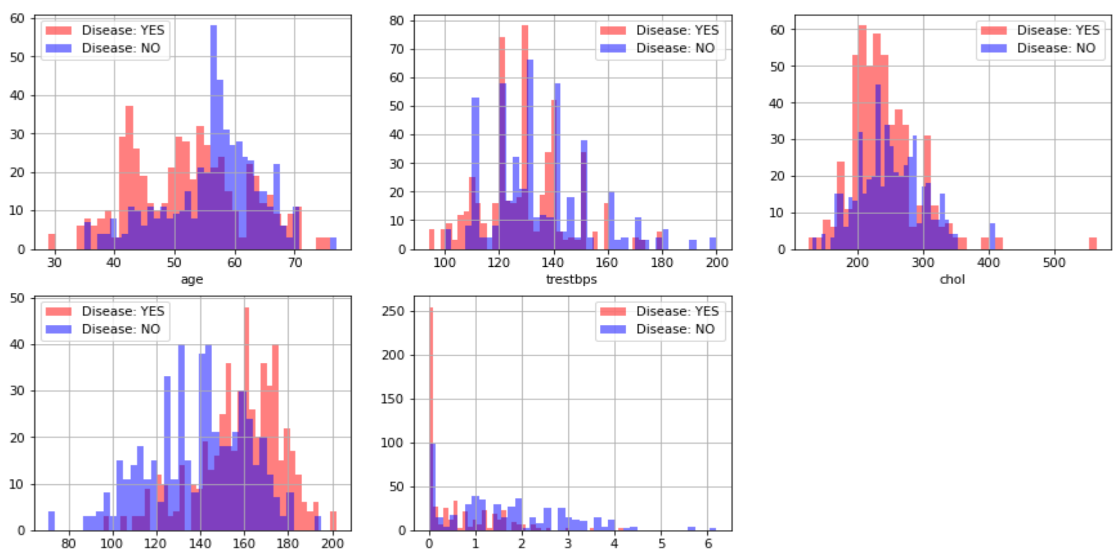
In [48]:
```
df['target'].value_counts()
```

Out[48]:
```
1    526
0    499
Name: target, dtype: int64
```

In [9]:
```
cat_values = []
conti_values = []

for col in df.columns:
    if len(df[col].unique()) >= 10:
        conti_values.append(col)
    else:
        cat_values.append(col)

print("catageroy values: ", cat_values)
print("continous values: ", conti_values)
```

```
catageroy values:  ['sex', 'cp', 'fbs', 'restecg', 'exang', 'slope', 'ca', 'thal', 'target']
continous values:  ['age', 'trestbps', 'chol', 'thalach', 'oldpeak']
```

In [10]:
```
plt.figure(figsize=(15,8))

for i, col in enumerate(conti_values, 1):
    plt.subplot(2,3,i)
    df[df.target ==1][col].hist(bins=40, color='red', alpha=0.5,  label='Disease: YES')
    df[df.target ==0][col].hist(bins=40, color='blue', alpha=0.5,  label='Disease: NO')
    plt.xlabel(col)
    plt.legend()
```

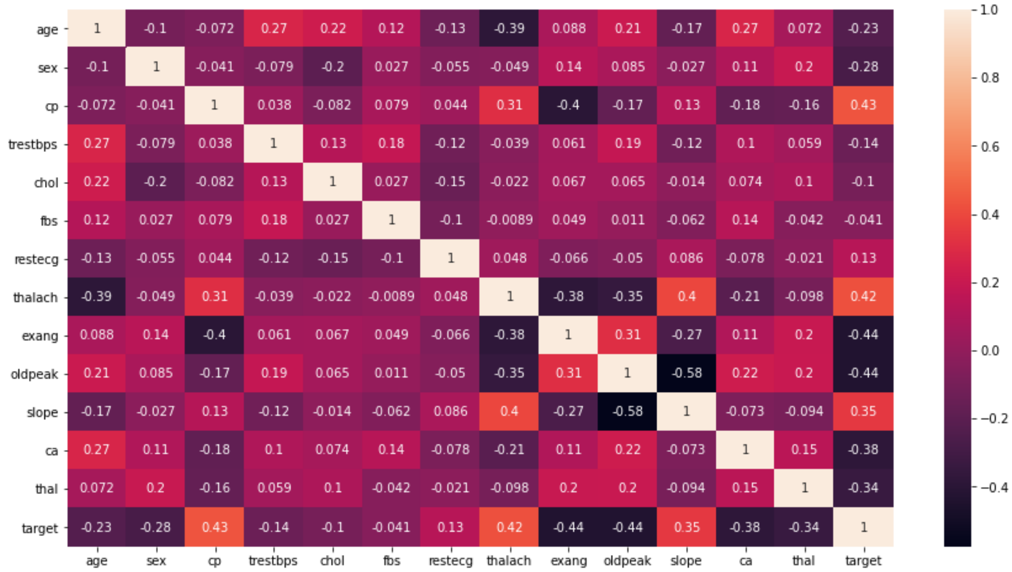



```
* trestbps[resting bp] anything above 130-140 is generally of concern
* chol[cholesterol] greater than 200 is of concern
* thalach People over 140 value are more likely to have heart disease
* oldpeak with value 0 are more than likely to have heart disease than any other value
```

Checking Correlation using Heatmap

In [11]:
```
x = df.corr()
plt.figure(figsize = (15,8))
sns.heatmap(x,annot = True)
```

Out[11]: <AxesSubplot:>



1. It is clearly visible that no column is a significant contributor among all the features.
2. So we are going to take all the features for the model evaluation.

7. Applicable Regulations

a. Patents on ML algorithms developed
b. Laws related to privacy for collecting data from users
c. Protection/ownership regulations
d. Creating an e-mail service to mail the report to the patient and doctor.
e. Being responsible by design.
f. Ensuring open-source, academic and research community for an audit of Algorithms.
h. Review of existing work authority regulations.

8. Applicable Constraints:

A. Requires a lot of research to obtain universal dataset of heart disease patients in-order to provide more sophisticated and accurate results.
B. Establishing e-mail service in the product which have to send the report after the machine learning model is deployed in any server.
C. Confidential health data to be obtained to train the model.
D. Thorough understanding of dataset and verification of the results must be performed by the pathologist from the machine learning model to provide a great health prescription and service to the user.

9. Business Opportunity

Pathologists are pretty good in diagnosing heart diseases while they are not so good in the prognosis of heart diseases.
It takes more than two weeks to identify heart disease in an individual. To overcome this hazardous
circumstance, our main objective is to use Machine Learning, which not only gives faster results but also demonstrates higher accuracy in the heart disease prediction process.

10. Concept Generation

This product requires the tool of machine learning models to be written from scratch in order to suit our needs. Tweaking these models for our use is less daunting than coding it up from scratch. A well-trained model can either be repurposed or built. But building a model with the resources and data we have is dilatory but possible. The customer might want to spend the least amount of time giving input data. This accuracy will take a little effort to nail because it's imprudent to rely purely on Classic Machine Learning algorithm.

1. First we clean the data

2. Split the data in x, y variable and Train_Test_Split the Data in X_train, X_test, y_train, y_test

Train - Test Split

```
In [18]: X = cleaned_data.drop(columns = 'target')
         y = cleaned_data['target']
```

```
In [19]: from sklearn.model_selection import train_test_split
         X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20, random_state=1)
```

3. Scaling the Data

Scaling

```
In [20]: from sklearn.preprocessing import StandardScaler
         sc = StandardScaler()
         X_train[conti_values] = sc.fit_transform(X_train[conti_values])
         X_test[conti_values] = sc.transform(X_test[conti_values])
```

We will use five different models and we will finalize the model which will give good accuracy

1. Logistic Regression

Applying Logistic Regression

In [21]:
```
from sklearn.linear_model import LogisticRegression
logreg = LogisticRegression()
logreg.fit(X_train, y_train)
```

Out[21]:
```
▾ LogisticRegression
LogisticRegression()
```

In [22]:
```
y_pred_test = logreg.predict(X_test)
```

In [23]:
```
from sklearn.metrics import accuracy_score, confusion_matrix
```

In [63]:
```
lr_acc_score=accuracy_score(y_test, y_pred_test)
lr_acc_score
```

Out[63]:
```
0.8673469387755102
```

Our model is **86.73 %** accurate by applying Logistic regeression

2. Naive Bayes

In [66]:
```
m2 = 'Naive Bayes'
nb = GaussianNB()
nb.fit(X_train,y_train)
nbpred = nb.predict(X_test)
nb_acc_score = accuracy_score(y_test, nbpred)
print(nb_acc_score)
```

```
0.8418367346938775
```

Our model is **84.18 %** accurate by applying Naive Bayes

3. Random Forest

In [67]:
```
m3 = 'Random Forest Classfier'
rf = RandomForestClassifier(n_estimators=20, random_state=12,max_depth=5)
rf.fit(X_train,y_train)
rf_predicted = rf.predict(X_test)
rf_acc_score = accuracy_score(y_test, rf_predicted)
print(rf_acc_score)
```

```
0.9285714285714286
```

Our model is **92.86 %** accurate by applying Random Forest Classfier

4. K-Nearest Neighbour

In [74]:
```
m4= 'K-Neighbors Classifier'
knn = KNeighborsClassifier(n_neighbors=10)
knn.fit(X_train, y_train)
knn_predicted = knn.predict(X_test)
knn_acc_score = accuracy_score(y_test, knn_predicted)
print(knn_acc_score)
```

```
0.8826530612244898
```

Our model is **88.27 %** accurate by applying K-Neighbors Classifier

5. Decision Tree

In [75]:
```
m5 = 'Decision Tree Classifier'
dt = DecisionTreeClassifier(criterion = 'entropy',random_state=0,max_depth = 6)
dt.fit(X_train, y_train)
dt_predicted = dt.predict(X_test)
dt_acc_score = accuracy_score(y_test, dt_predicted)
print(dt_acc_score)
```

```
0.9387755102040817
```

Our model is **93.88 %** accurate by applying Decision Tree Classifier

Model Evaluation in percentage

```
In [73]: model_ev = pd.DataFrame({'Model': ['Logistic Regression','Naive Bayes','Random Forest',
                            'K-Nearest Neighbour','Decision Tree'], 'Accuracy': [lr_acc_score*100,
                            nb_acc_score*100,rf_acc_score*100,knn_acc_score*100,dt_acc_score*100]})
         model_ev
```

Out[73]:

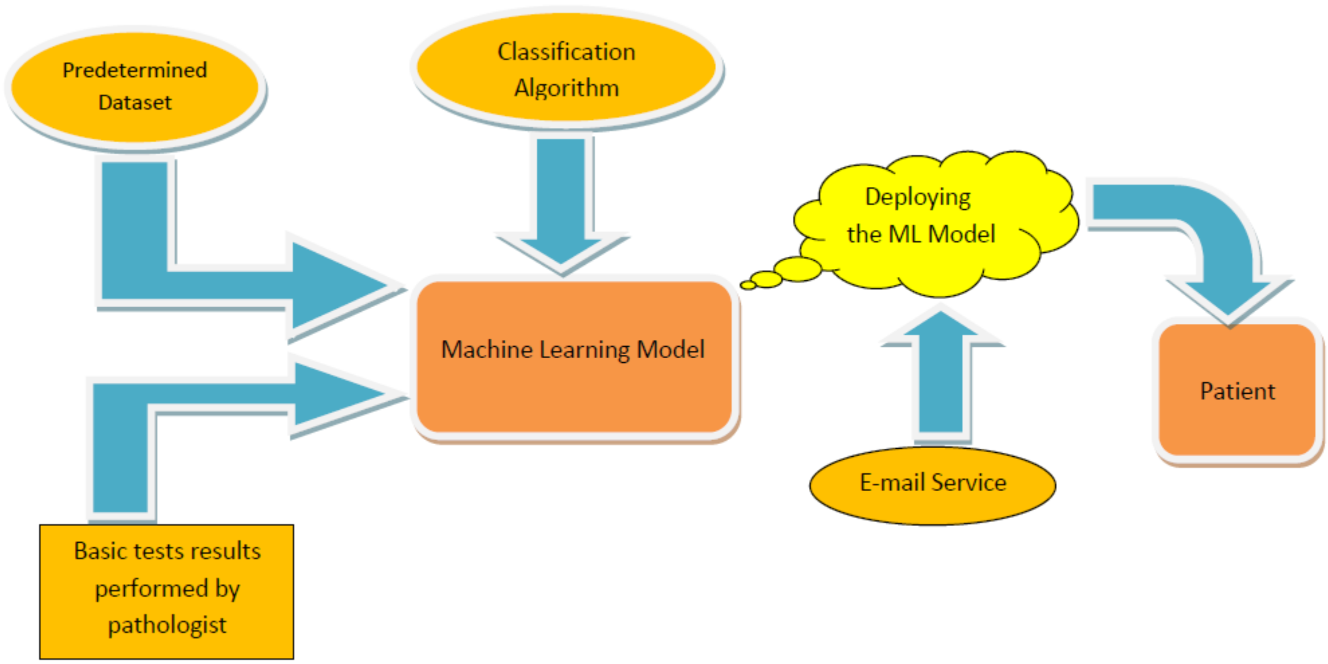
	Model	Accuracy
0	Logistic Regression	86.734694
1	Naive Bayes	84.183673
2	Random Forest	92.857143
3	K-Nearest Neighbour	88.265306
4	Decision Tree	93.877551

Over all the Machine Learning Algorithms, **Decision Tree(93.88 %)** and **Random Forest(92.86 %)** Algorithm gives us the best Accuracy.

11. Concept Development

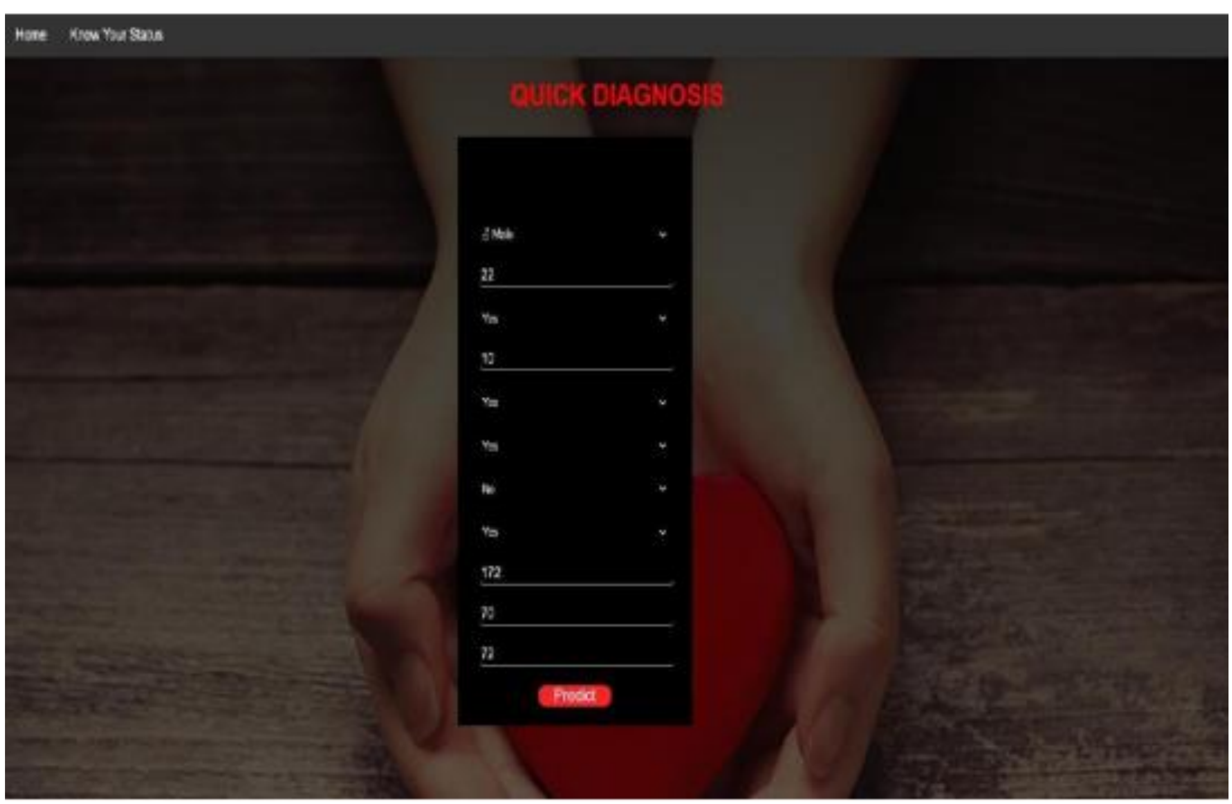
The concept can be developed by using the appropriate API (flask in this case) and using Django as framework for the same and for its deployment, the cloud services must be chosen accordingly to the need

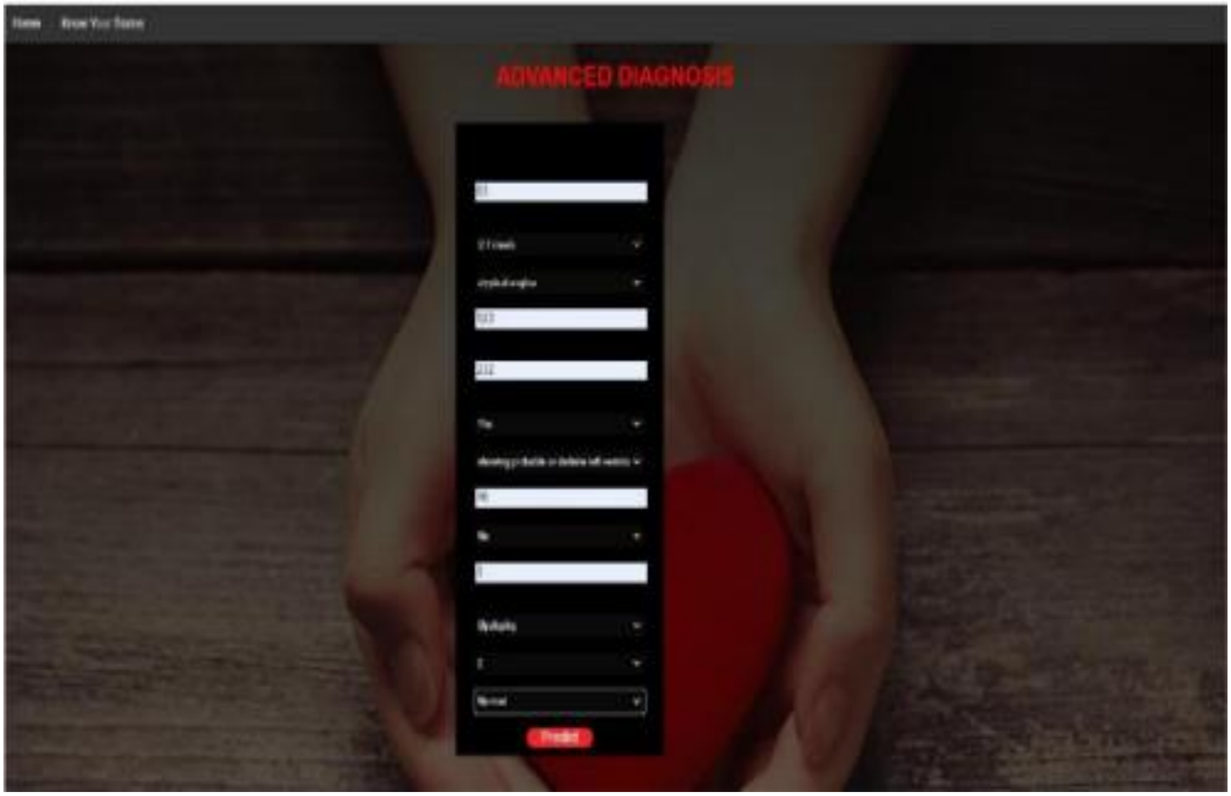
12. Final Product Prototype



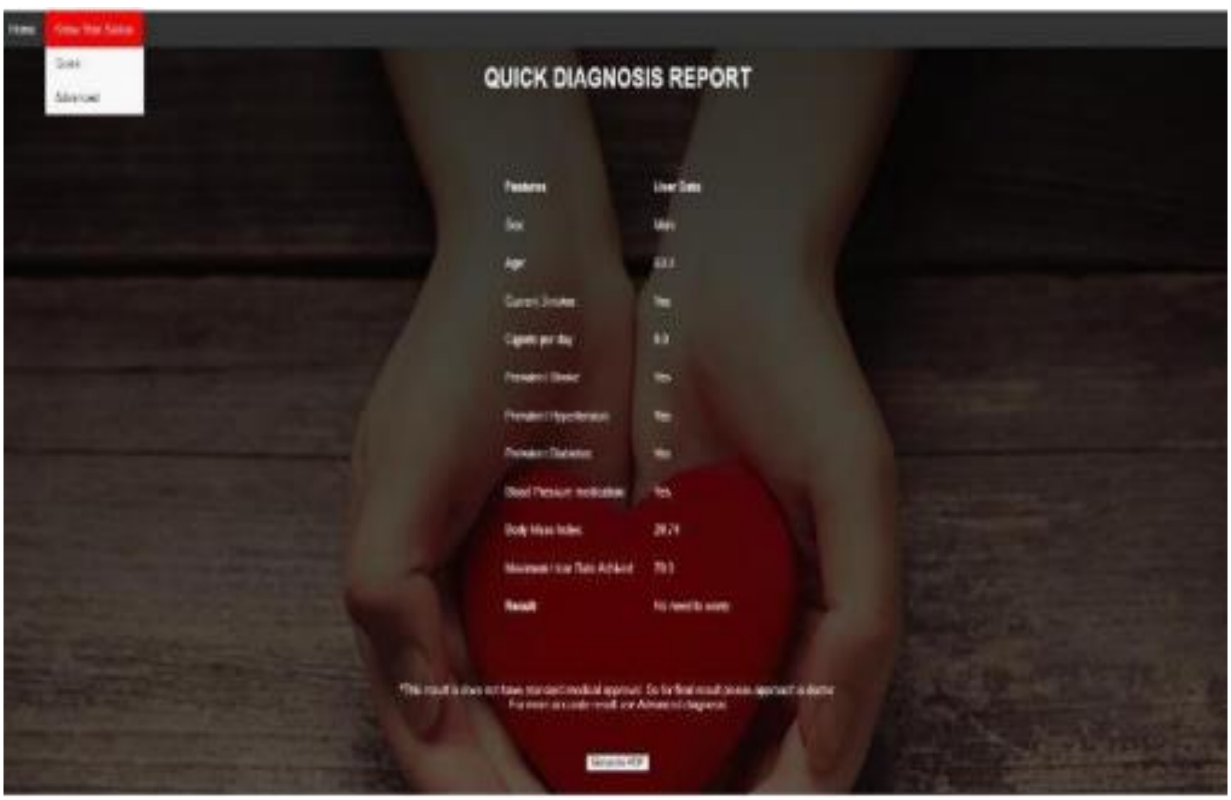
13. Product details

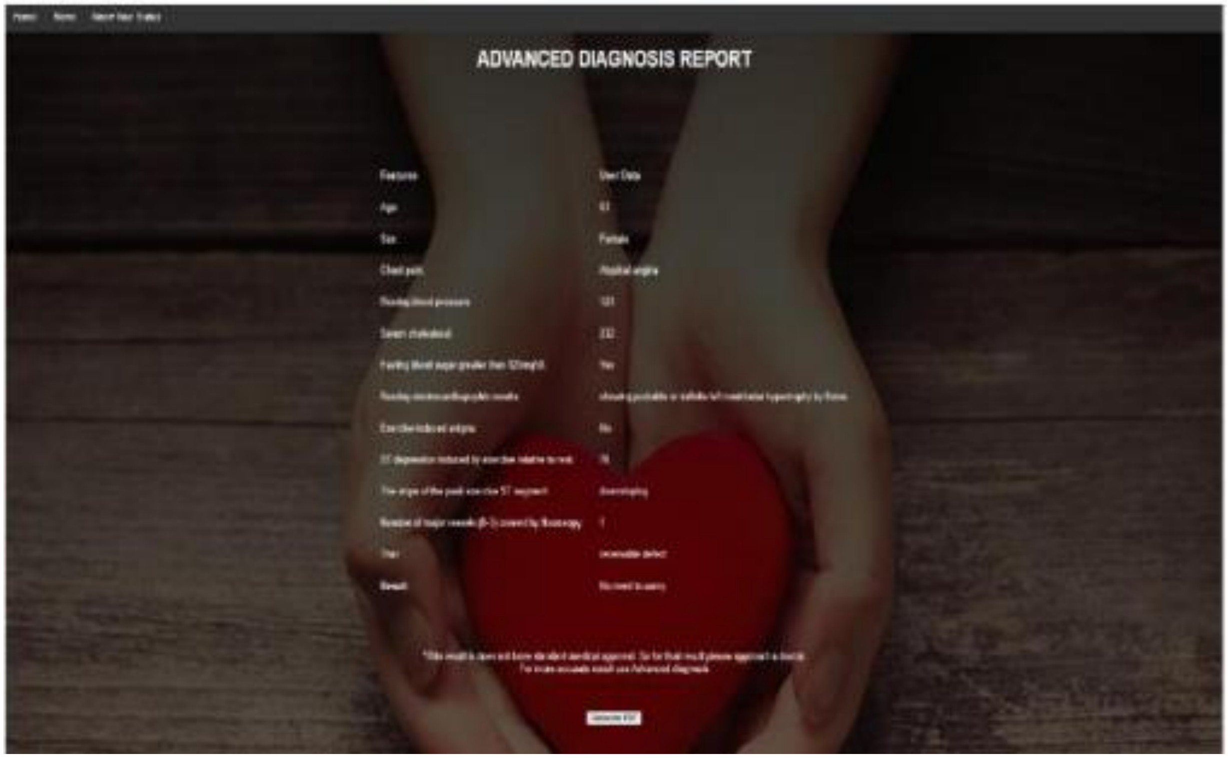
Input





output





14. Code Implementation

Kaggle Link:

https://www.kaggle.com/virajparab1562/heart-disease-prediction

15. Conclusion

AI is set to change the medical industry in the coming decades — it wouldn't make sense for pathology to not be disrupted too. Currently, ML models are still in the testing and experimentation phase for heart disease prognoses. As datasets are getting larger and of higher quality, researchers are building increasingly accurate models. While we might not see AI doing the job of a pathologist today, we can expect ML to replace our local pathologist in the coming decades, and it's exciting! ML models still have a long way to go, most models still lack sufficient data and suffer from bias. Machine learning can train just as well as doctor prognosis, it doesn't require extra pay for prognosis. Manual heart disease treatment takes long time to show the result, while machine learning gives output in seconds. To save people's life and allow doctor to fully concentrate in diagnosis, yet something we are certain of is that ML is the next step of pathology, and it will disrupt the industry.