IoT approach for Heart Disease Prediction

By

Siddharth Khachane (18BCE1013)
Harsh Kailash (18BCE1340)

A project report submitted to

Prof. Rekha D

SCHOOL OF COMPUTING SCIENCE AND ENGINEERING

in partial fulfilment of the requirements for the course of

ECE3502 – IoT Domain Analyst

in

B.Tech. COMPUTER SCIENCE AND ENGINEERING

VIT
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

VIT CHENNAI

Vandalur – Kelambakkam Road

Chennai – 600127

MAY 2021

BONAFIDE CERTIFICATE

Certified that this project report entitled "**IoT Approach for Heart Disease Prediction"** is a bonafide work of **Siddharth Khachane (18BCE1013) and Harsh Kailash (18BCE1340)** who carried out the Project work under my supervision and guidance for **ECE3502 – IoT Domain Analyst**

Prof. Rekha D

Assistant Professor

School of Electronics Engineering (SENSE),

VIT Chennai

Chennai – 600 127.

ABSTRACT

This project is developed by using different machine learning algorithm on Heart Disease datasets and predicting, analyzing and visualizing dataset. The coding and visualization is done in python.

There are various techniques that we have used to predict heart disease including classification algorithms such as Random Forest and XGBoost as well as neural network algorithms.

We have made an IoT prototype to generate a Heart disease dataset by simulating different IoT sensors in Node-Red.

We have also collected unstructured data from different users by circulating heart disease and we have also analyzed it.

ACKNOWLEDGEMENT

We wish to express our sincere thanks and deep sense of gratitude to our project guide, **Prof. Rekha D,** Assistant Professor, School of Computer Science, for her consistent encouragement and valuable guidance offered to us in a pleasant manner throughout the course of the project work.

We also take this opportunity to thank all the faculty of the School for their support and their wisdom imparted to us throughout the course.

We thank our parents, family, and friends for bearing with us throughout the course of our project and for the opportunity they provided us in undergoing this course in such a prestigious institution.

TABLE OF CONTENTS

S. No.	Topic	Page
	Abstract	3
	Acknowledgement	4
1	Introduction	6
2	About the Dataset	7
3	Algorithm	8
4	Generated structured data	10
5	Generated unstructured data	12
6	Software Implementation	14
7	Displaying the prediction	23
8	Comparison of Results	25
9	Conclusion	26
10	PPT for rev 3	27

1. INTRODUCTION

Heart disease has become more common now-a-days due to unhealthy eating habits and increasing amount of stress each day. Heart disease can be cured relatively easily if it's detected in initial stages. However, it's difficult to predict whether a particular person is showing symptoms accurately without the help of algorithms. So we have trained different models using different machine learning algorithms to predict whether a person is showing the signs of heart disease.

We have created a prototype for generating values for structured data in nodered which is further analyzed on python.

We have carried this project one step ahead by including prediction on Unstructured data collected using Google form

2. DATASET:

Age: displays the age of the individual.

Sex: displays the gender of the individual using the following format:

- 1 = male
- 0 = female

Chest-pain type: displays the type of chest-pain experienced by the individual using the following format :

- 1 = typical angina
- 2 = atypical angina
- 3 = non — anginal pain
- 4 = asymptotic

Resting Blood Pressure: displays the resting blood pressure value of an individual in mmHg (unit)

Serum Cholestrol: displays the serum cholesterol in mg/dl (unit)

Fasting Blood Sugar: compares the fasting blood sugar value of an individual with 120mg/dl.

If fasting blood sugar > 120mg/dl then : 1 (true)

else : 0 (false)

Resting ECG : displays resting electrocardiographic results

- 0 = normal
- 1 = having ST-T wave abnormality
- 2 = left ventricular hyperthrophy

Max heart rate achieved : displays the max heart rate achieved by an individual.

Exercise induced angina :

- 1 = yes
- 0 = no

ST depression induced by exercise relative to rest: displays the value which is an integer or float.

Peak exercise ST segment :

- 1 = upsloping
- 2 = flat
- 3 = downsloping

Number of major vessels (0–3) colored by flourosopy : displays the value as integer or float.

Thal : displays the thalassemia :

- 3 = normal
- 6 = fixed defect
- 7 = reversible defect

Diagnosis of heart disease : Displays whether the individual is suffering from heart disease or not :

- 0 = absence
- 1, 2, 3, 4 = present.

3. ALGORITHMS:

Random Forest:

Random Forest is based on the principle of decision trees.

So basically, we provide the number of decision trees that should be formed and all these decision trees would give their predictions. Now this is extremely beneficial because a large number of relatively uncorrelated models (trees) operating as a committee will outperform any of the individual constituent models.

Also, these different decision trees should be as uncorrelated with each other as possible as Uncorrelated models can produce ensemble predictions that are more accurate than any of the individual predictions. The reason for this wonderful effect is that the trees protect each other from their individual errors.

Some of its benefits are:

1. It takes less training time as compared to other algorithms.
2. It predicts output with high accuracy, even for the large dataset it runs efficiently.
3. It can also maintain accuracy when a large proportion of data is missing.

XGBoost:
The use of the algorithm was for efficiency of compute time and memory resources. A design goal was to make the best use to train the model. Some key algorithm implementation features include:

Some of its benefits are

1) Sparse Aware: missing data values are automatically recognised and removed.
2) Block Structure: to enhance the parallelization of tree construction.
3) Continued Training: so that we can further boost an already fitted model on new data.

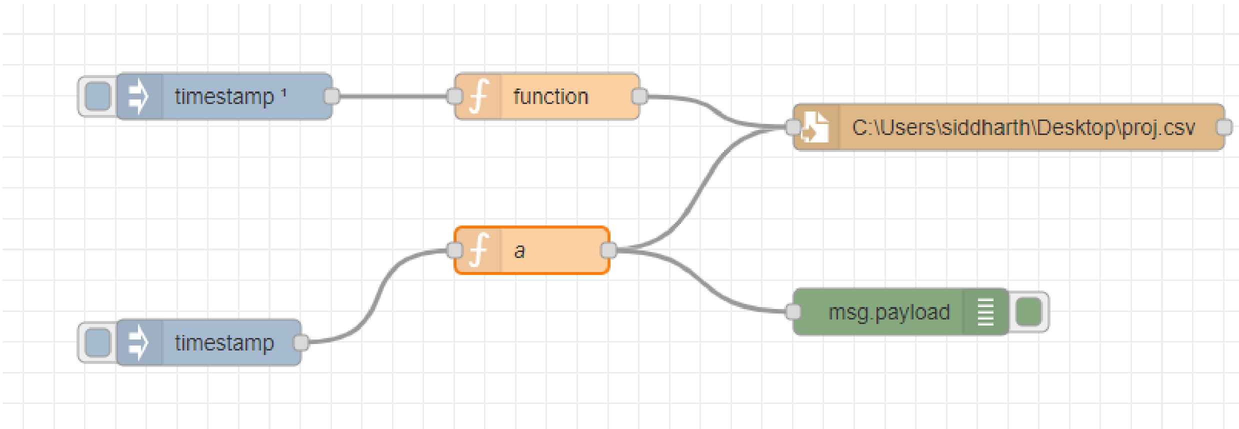
Neural Network:
Neural Network is basically a chain of algorithms in which the neurons try to mimic the human brain in order to find relationships in the dataset.

It consists of various layers:

- Input layer — it accepts the initial data for the neural network.
- Hidden layers — they are the intermediate layer between input and output layer and place where all the computation is done.
- Output layer — produce the result for given inputs.

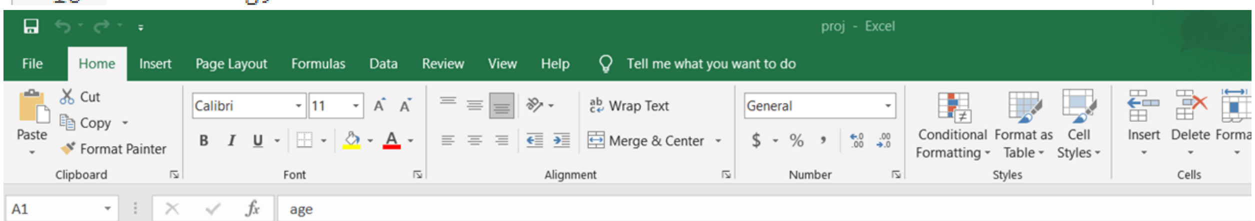
4. GENERATED STRUCTURED DATA:

We have come up with an IoT prototype in Node-red for simulating the sensors and generating the values for heart disease prediction.




```
var a='age';
var b='gender';
var c='cp';
var d='tre';
var e='chol';
var f='fbs';
var g='rest';
var h='thalch';
var i='exang';
var j="oldpeak";
var k="slope";
var l="ca";
var m="thal";

msg.payload=a+","+b+","+c+","+d+","+e+","+f+","+g+","+h+","+i+","+j+","
return msg;
```

```
var a=Math.round(Math.random()*48)+29;
var b=Math.round(Math.random()*1);
var c=Math.round(Math.random()*3);
var d=Math.round(Math.random()*106)+94;
var e=Math.round(Math.random()*438)+126;
var f=Math.round(Math.random()*1);
var g=Math.round(Math.random()*2);
var h=Math.round(Math.random()*131)+71;
var i=Math.round(Math.random()*1);
var j=Math.round(Math.random()*6);
var k=Math.round(Math.random()*2);
var l=Math.round(Math.random()*4);
var m=Math.round(Math.random()*3);
msg.payload=a+","+b+","+c+","+d+","+e+","+f+","+g+","+h+","+i+","+j+","

return msg;
```

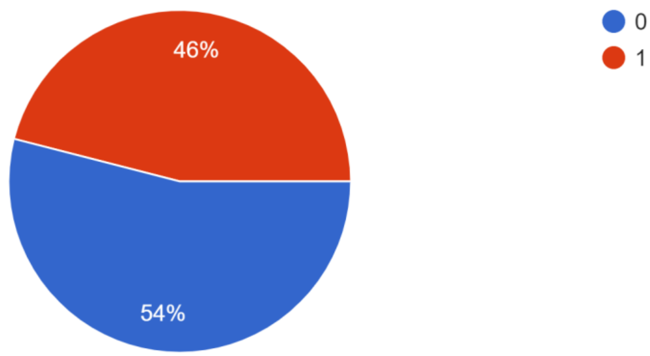


	A	B	C	D	E	F	G	H	I	J	K	L	M
1	age	gender	cp	tre	chol	fbs	rest	thalch	exang	oldpeak	slope	ca	thal
2	38	1	1	186	248	0	2	172	0	3	2	1	1
3	56	0	3	165	523	0	1	179	1	3	2	2	1
4	49	0	1	198	156	1	1	97	1	0	1	0	1
5	43	0	2	162	303	1	0	94	0	4	1	2	0
6	40	1	0	188	182	1	2	156	0	5	1	3	1
7	43	1	1	164	287	0	2	123	1	1	1	3	2
8	34	1	2	97	282	0	2	131	1	1	1	0	2
9	31	1	3	97	514	0	2	150	0	4	1	1	2
10	68	1	1	148	439	1	1	153	0	4	0	4	1
11	44	0	2	191	164	1	1	75	0	1	2	1	1

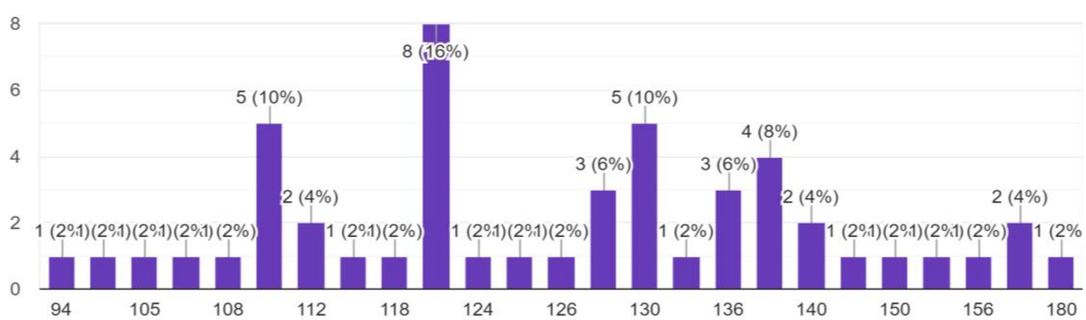
5. GENERATED UNSTRUCTURED DATA:

We conducted a experimental study (survey) for collection of some attributes related to heart disease. It contains 5 attributes namely

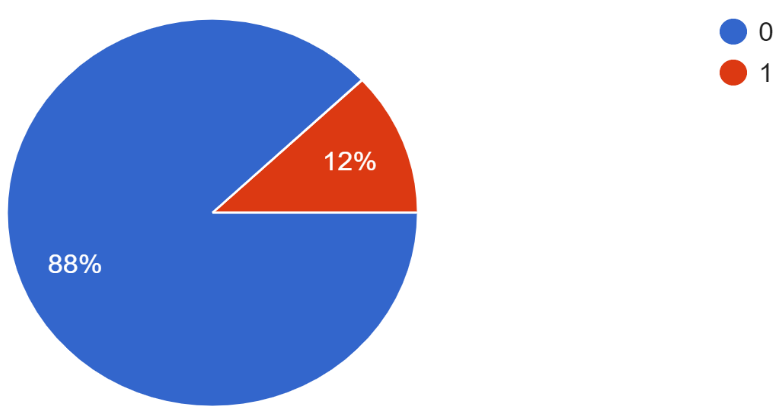
Sex
50 responses



Blood Pressure
50 responses



Blood Sugar
50 responses



B	C	D	E	F	G
Age	Sex	Blood Pre	Cholester	Blood Sug	Target
69	1	160	234	1	1
45	0	138	236	0	1
50	0	120	244	0	1
50	0	110	254	0	1
64	0	180	325	0	1

6. SOFTWARE IMPLEMENTATION:

Random Forest:

Reading the data

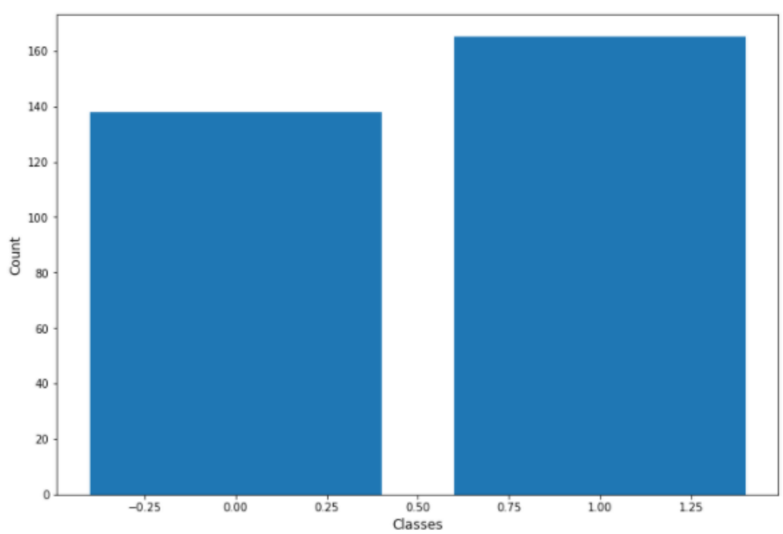
```
df = pd.read_csv('/kaggle/input/heart-disease-uci/heart.csv')
df.shape
```

```
(303, 14)
```

```
df.head()
```

	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

```
unique_class, counts_class = np.unique(y, return_counts=True)
fig = plt.figure()
ax = fig.add_axes([0,0,1.5,1.5])
ax.bar(unique_class,counts_class)
ax.set_xlabel('Classes', fontsize='large')
ax.set_ylabel('Count', fontsize='large')

plt.show()
```



Training the model on different tree sizes

```
time_rf_list=[]
rf_accuracy=[]
y_pred_list=[]

for n in tree_list:

    rf_clf = RandomForestClassifier(n_estimators=n, random_state=121,criterion='entropy')

    rf_clf.fit(X_train, y_train)

    y_pred_list.append(rf_clf.predict(X_test))

    rf_accuracy.append(metrics.accuracy_score(y_test,rf_clf.predict(X_test)))
```

```
rf_accuracy
```

```
[0.8360655737704918,
 0.8524590163934426,
 0.8360655737704918,
 0.8360655737704918]
```

Finally training the model on the optimal (n_estmators=50)

```
from sklearn.ensemble import RandomForestClassifier
model = RandomForestClassifier(n_estimators=50)
model.fit(X_train, y_train)
Y_pred = model.predict(X_test)

from sklearn.metrics import classification_report
print(classification_report(y_test,Y_pred))
```

```
              precision    recall  f1-score   support

           0       0.88      0.85      0.87        27
           1       0.89      0.91      0.90        34

    accuracy                           0.89        61
   macro avg       0.89      0.88      0.88        61
weighted avg       0.89      0.89      0.88        61
```

XGBoost:

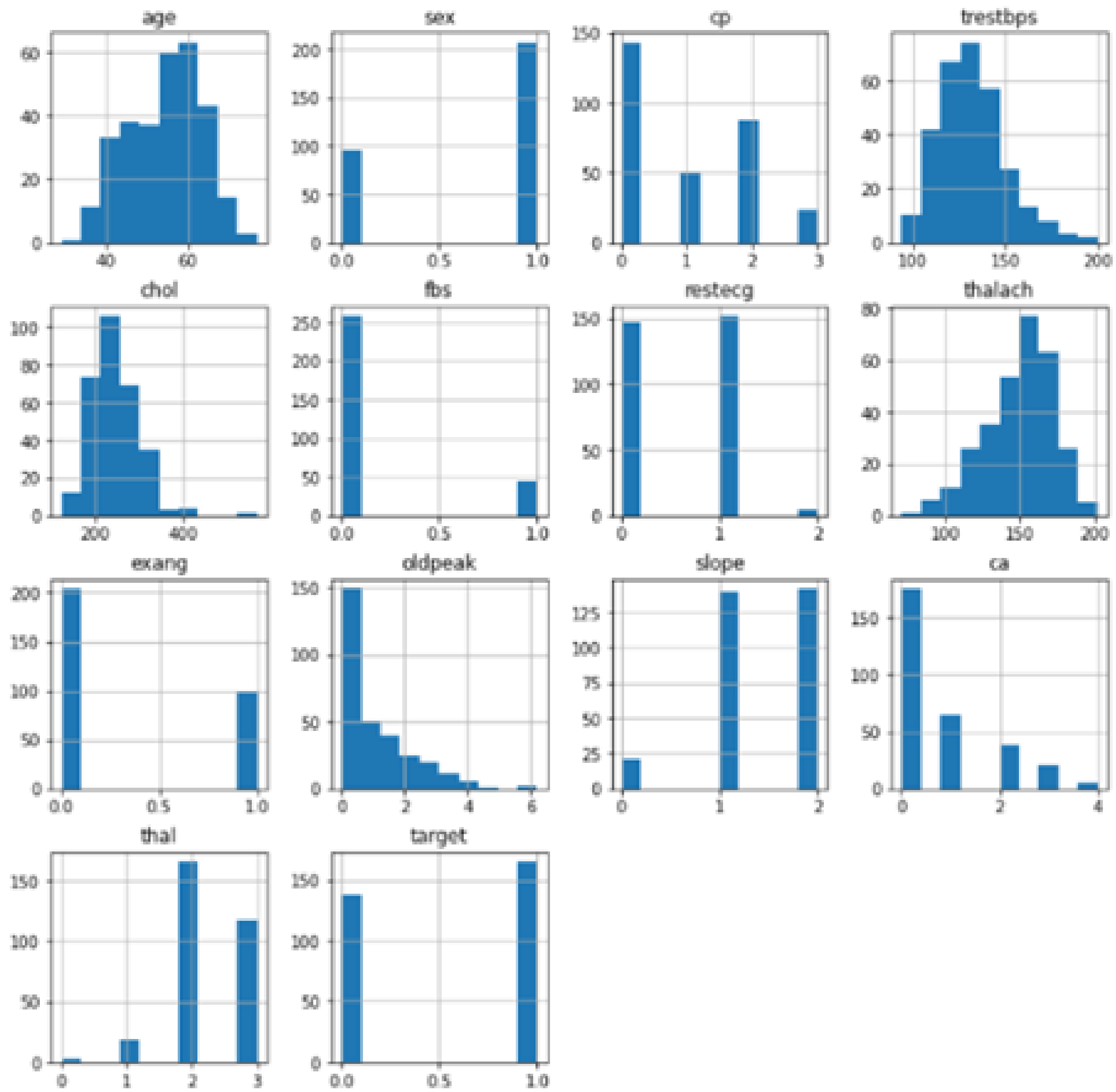
```
print("----XGB-----")
from sklearn.metrics import accuracy_score
model = XGBClassifier()
model.fit(x_train, y_train)
y_pred = model.predict(x_test)
predictions = [round(value) for value in y_pred]
accuracy = accuracy_score(y_test, predictions)
print("Accuracy: %.2f%%" % (accuracy * 100.0))
```

```
----XGB-----
[21:53:12] WARNING: C:/Users/Administrator/worksp
inary:logistic' was changed from 'error' to 'logl
Accuracy: 86.89%
```

Neural Network (Structured data)

```
import sys
import pandas as pd
import numpy as np
import sklearn
import matplotlib
import keras
import matplotlib.pyplot as plt
from pandas.plotting import scatter_matrix
import seaborn as sns
from keras.models import Sequential
from keras.layers import Dense
from keras.optimizers import Adam
from keras.layers import Dropout
from keras import regularizers
```

```
df.hist(figsize = (12, 12))
plt.show()
```



```
X = np.array(df.drop(['target'], 1))
y = np.array(df['target'])
```

```
mean = X.mean(axis=0)
X -= mean
std = X.std(axis=0)
X /= std
```

```
from sklearn import model_selection

X_train, X_test, y_train, y_test = model_selection.train_test_split(X, y, stratify=y, random_st
ate=42, test_size = 0.2)
```

```
from keras.utils.np_utils import to_categorical

Y_train = to_categorical(y_train, num_classes=None)
Y_test = to_categorical(y_test, num_classes=None)
print (Y_train.shape)
print (Y_train[:10])
```

```
(242, 2)
[[0. 1.]
 [1. 0.]
 [1. 0.]
 [1. 0.]
 [0. 1.]
 [0. 1.]
 [0. 1.]
 [0. 1.]
 [1. 0.]
 [0. 1.]]
```

Neural network on unstructured data

```
# Import packages that we will be working with.
import os
import numpy as np
import pandas as pd
from keras.layers import Dense
import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow.python.keras.models import Sequential
from tensorflow.python.keras.layers import Dense
from keras.optimizers import Adam
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, accuracy_score
```

```
# Load the dataset, and view couple of the first rows.
data = pd.read_csv("heart1.csv")
print(data.head(3))

# Check the datatypes
print(data.dtypes)
```

```
   age  sex  trestbps  chol  fbs  target
0   69    1       160   234    1       1
1   45    0       138   236    0       1
2   50    0       120   244    0       1
age         int64
sex         int64
trestbps    int64
chol        int64
fbs         int64
target      int64
dtype: object
```

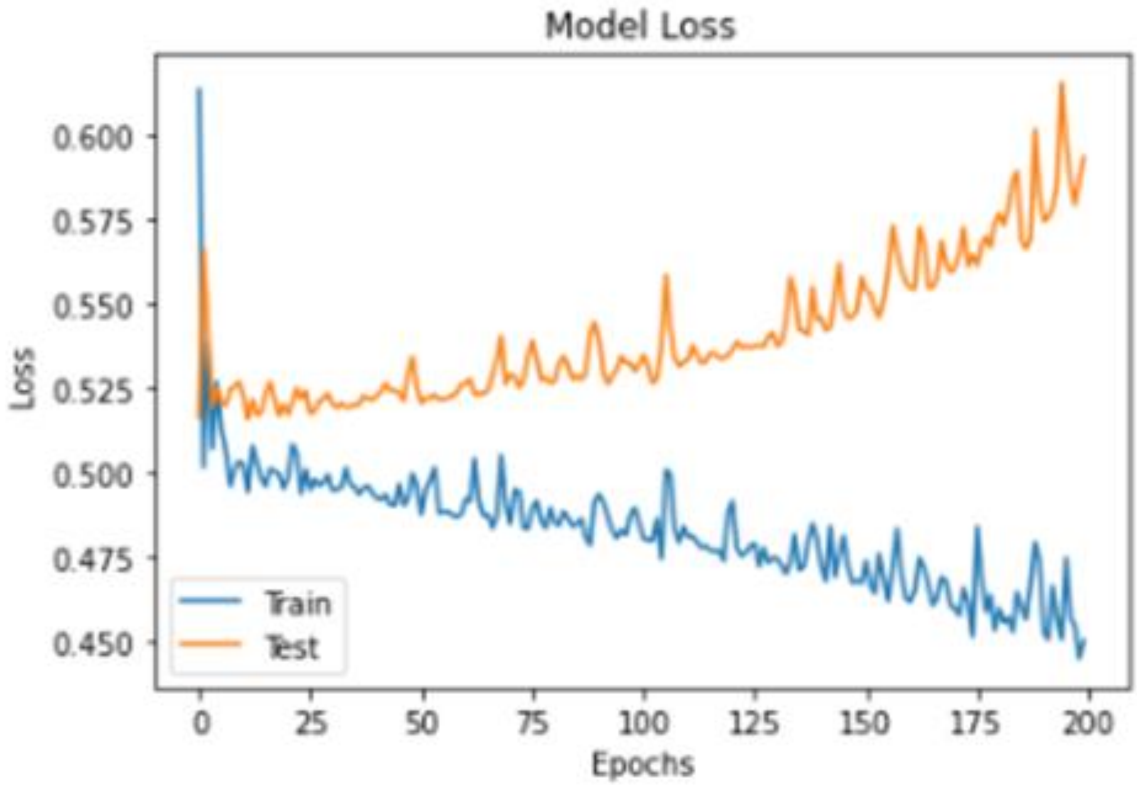
```
Y = data.target.values
X = data.drop(['target'], axis=1)

# Now split to train/test with 80% training data, and 20% test data.
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_state=42)

# Check dimensions of both sets.
print("Train Features Size:", X_train.shape)
print("Test Features Size:", X_test.shape)
print("Train Labels Size:", Y_train.shape)
print("Test Labels Size:", Y_test.shape)
```

```
Train Features Size: (40, 5)
Test Features Size: (10, 5)
Train Labels Size: (40,)
Test Labels Size: (10,)
```

```
history = model.fit(X_train, Y_train, validation_data=(X_test, Y_test), epochs=200, batch_size=16, verbose=2)
```

```
Epoch 1/200
3/3 - 3s - loss: 0.6136 - accuracy: 0.6500 - val_loss: 0.5165 - val_accuracy: 0.8000
Epoch 2/200
3/3 - 0s - loss: 0.5017 - accuracy: 0.8000 - val_loss: 0.5665 - val_accuracy: 0.8000
Epoch 3/200
3/3 - 0s - loss: 0.5386 - accuracy: 0.8000 - val_loss: 0.5418 - val_accuracy: 0.8000
Epoch 4/200
3/3 - 0s - loss: 0.5072 - accuracy: 0.8000 - val_loss: 0.5195 - val_accuracy: 0.8000
Epoch 5/200
3/3 - 0s - loss: 0.5269 - accuracy: 0.8000 - val_loss: 0.5251 - val_accuracy: 0.8000
Epoch 6/200
3/3 - 0s - loss: 0.5130 - accuracy: 0.8000 - val_loss: 0.5211 - val_accuracy: 0.8000
Epoch 7/200
3/3 - 0s - loss: 0.5067 - accuracy: 0.8000 - val_loss: 0.5200 - val_accuracy: 0.8000
Epoch 8/200
3/3 - 0s - loss: 0.4962 - accuracy: 0.8000 - val_loss: 0.5248 - val_accuracy: 0.8000
Epoch 9/200
3/3 - 0s - loss: 0.5010 - accuracy: 0.8000 - val_loss: 0.5255 - val_accuracy: 0.8000
Epoch 10/200
3/3 - 0s - loss: 0.5035 - accuracy: 0.8000 - val_loss: 0.5271 - val_accuracy: 0.8000
Epoch 11/200
3/3 - 0s - loss: 0.5025 - accuracy: 0.8000 - val_loss: 0.5231 - val_accuracy: 0.8000
Epoch 12/200
3/3 - 0s - loss: 0.4944 - accuracy: 0.8000 - val_loss: 0.5160 - val_accuracy: 0.8000
Epoch 13/200
3/3 - 0s - loss: 0.5079 - accuracy: 0.8000 - val_loss: 0.5217 - val_accuracy: 0.8000
Epoch 14/200
3/3 - 0s - loss: 0.5031 - accuracy: 0.8000 - val_loss: 0.5171 - val_accuracy: 0.8000
Epoch 15/200
3/3 - 0s - loss: 0.4983 - accuracy: 0.8000 - val_loss: 0.5180 - val_accuracy: 0.8000
Epoch 16/200
3/3 - 0s - loss: 0.4961 - accuracy: 0.8000 - val_loss: 0.5236 - val_accuracy: 0.8000
Epoch 17/200
3/3 - 0s - loss: 0.5012 - accuracy: 0.8000 - val_loss: 0.5269 - val_accuracy: 0.8000
Epoch 18/200
```

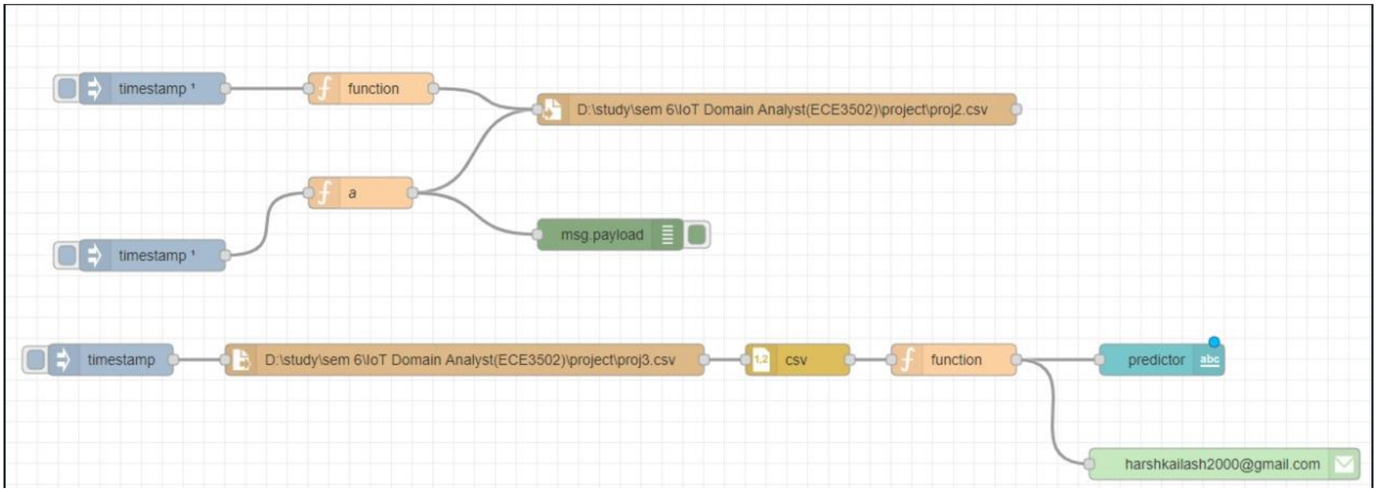
```
# Plot the Loss function vs. number of Epochs
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Model Loss')
plt.ylabel('Loss')
plt.xlabel('Epochs')
plt.legend(['Train', 'Test'])
plt.show()
```




```
predictions = np.argmax(model.predict(X_test), axis=1)
model_accuracy = accuracy_score(Y_test, predictions)*100
print("Model Accracy:", model_accuracy,"%")
print(classification_report(Y_test, predictions))
```

```
Model Accracy: 80.0 %
              precision    recall  f1-score   support

           0       0.00      0.00      0.00         2
           1       0.80      1.00      0.89         8

    accuracy                           0.80        10
   macro avg       0.40      0.50      0.44        10
weighted avg       0.64      0.80      0.71        10
```

7. DISPLAYING THE PREDICTION



USER INPUT CSV FILE

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	age	gender	cp	tre	chol	fbs	rest	thalch	exang	oldpeak	slope	ca	thal
2	58	0	0	126	489	1	0	172	0	1	2	4	2
3													

PYTHON CODE

```
[24]: import csv
```

```
[25]: df = pd.read_csv('proj2.csv')
      df
```

[25]:

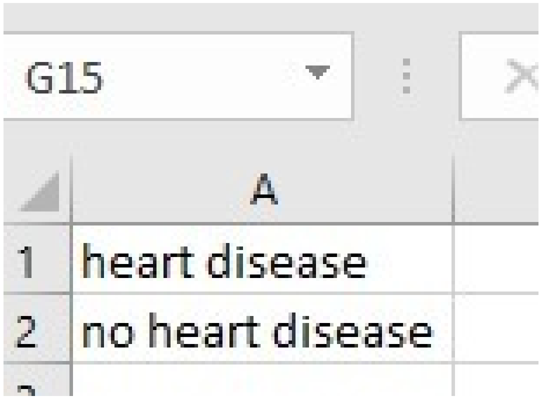
	age	gender	cp	tre	chol	fbs	rest	thalch	exang	oldpeak	slope	ca	thal
0	58	0	0	126	489	1	0	172	0	1	2	4	2

```
[27]: z=model.predict(df[0:])
```

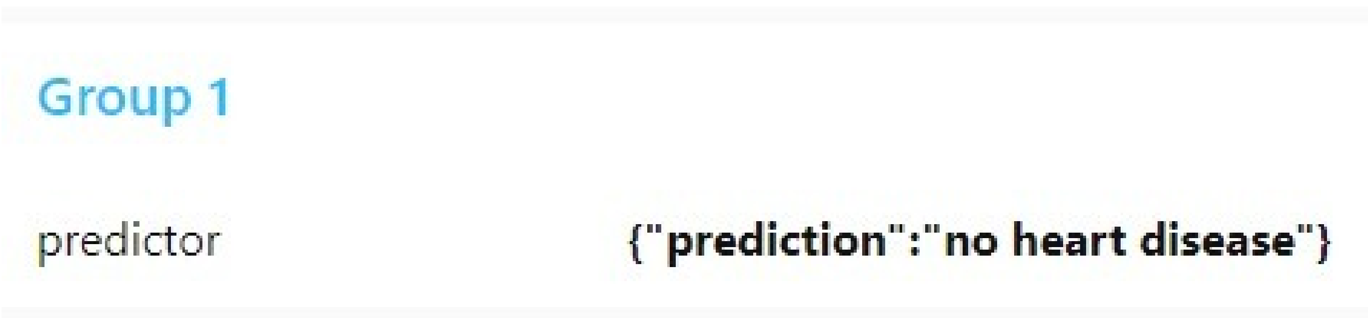
```
[28]: if(z[0]==0):
          str="no heart disease"
      else:
          str="heart disease"
```

```
[31]: field=['heart disease']
      row=[[str]]
      filename = "predictor.csv"
      with open(filename, 'w') as csvfile:
          # creating a csv writer object
          csvwriter = csv.writer(csvfile)

          # writing the fields
          csvwriter.writerow(field)

          # writing the data rows
          csvwriter.writerows(row)
```

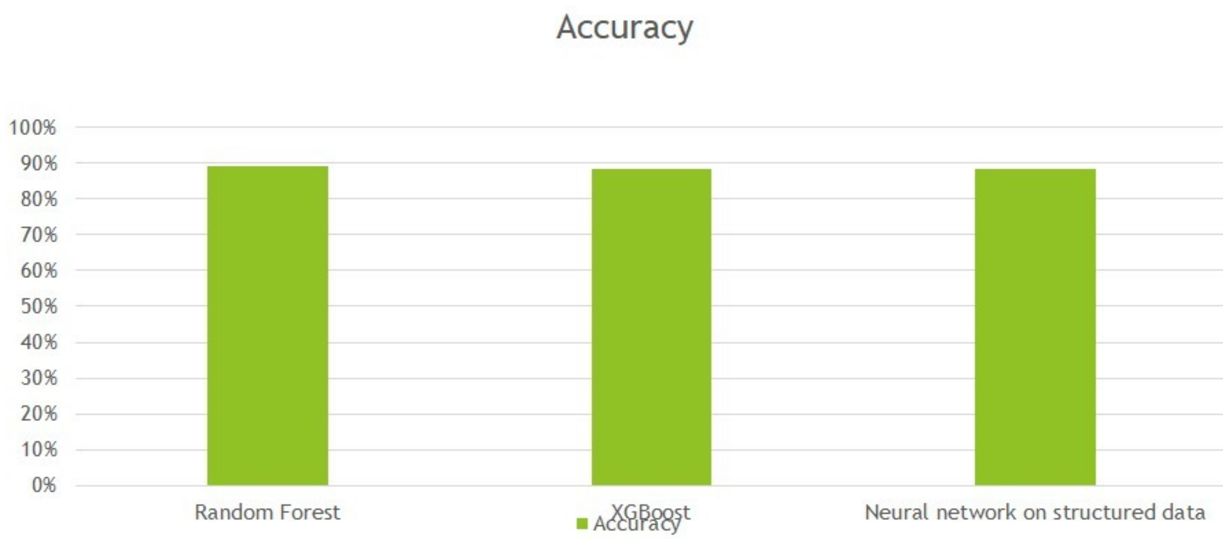
PREDICTION CSV FILE



DASHBOARD



8. COMPARISION:



	Accuracy
Random Forest	89%
XGBoost	88.52%
Neural network on structured data	88.50%

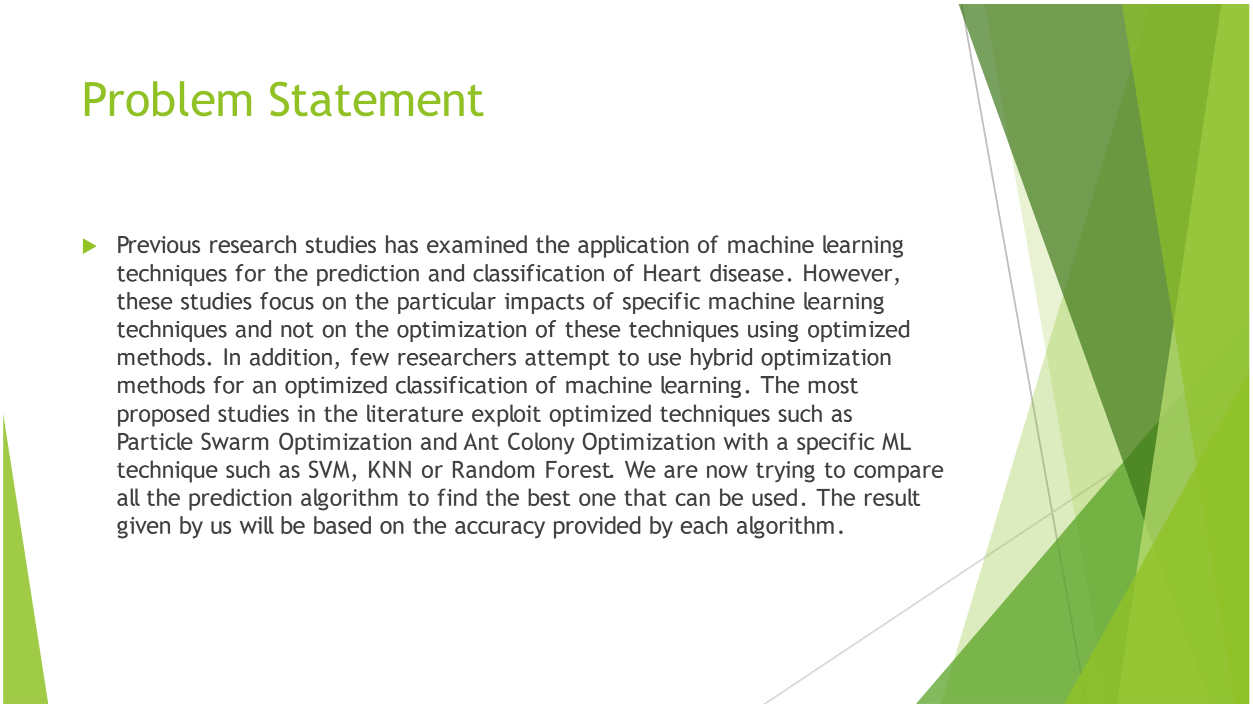
These models were implemented on the structured dataset which is the same for all of them.

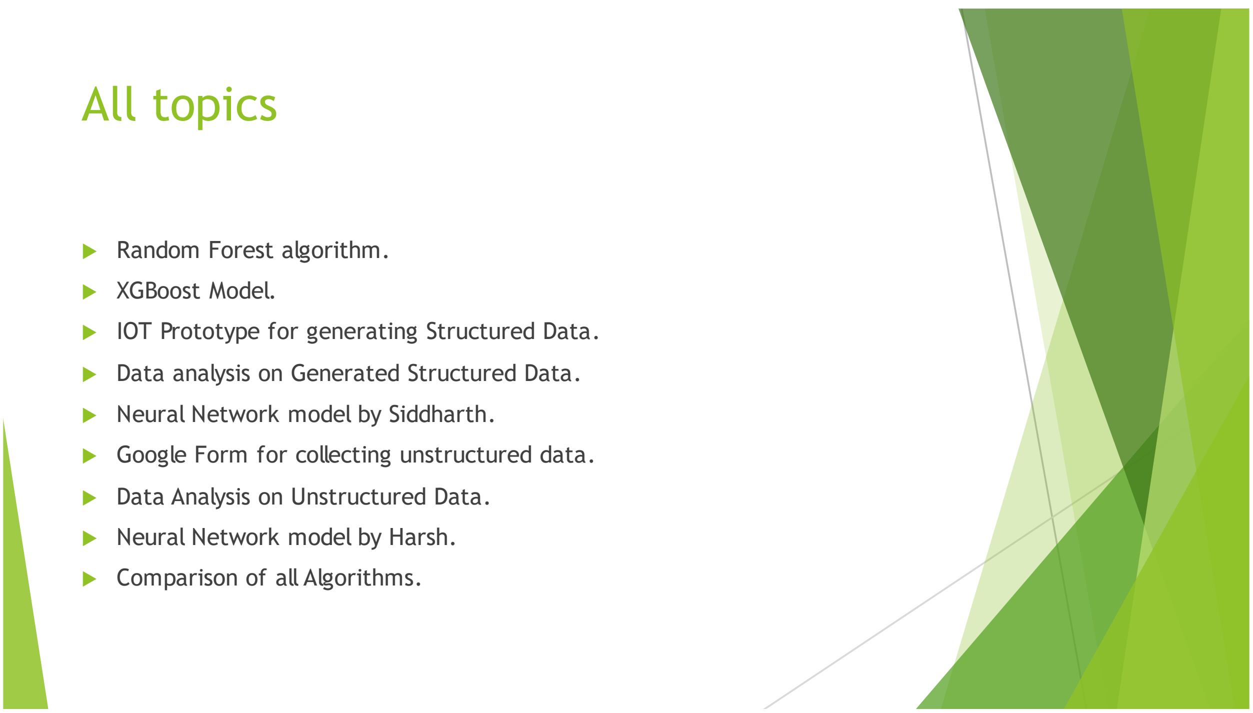
But for unstructured the dataset was different so it is not included in the comparison.
For unstructured we achieved 80% accuracy.

9. CONCLUSION:

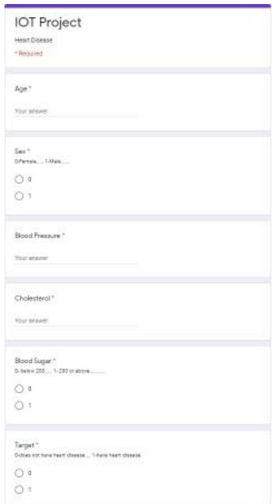
- We have covered normal classification algorithms like the Random forest and the XGBoost and also neural network algorithms.
- Since the data for this problem was normal numerical data the neural networks didn't have a clear cut advantage which they usually have in case of images.
- Thus they performed almost the same on structured data.
- We have also included a prediction system in Node-red where the new values will be predicted by the model and the predictions will be displayed on the dashboard.

10. Review 3 ppt

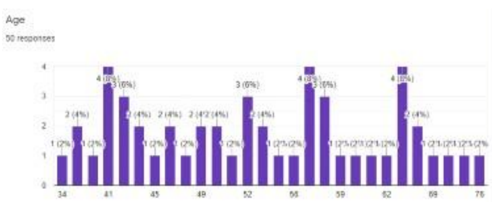


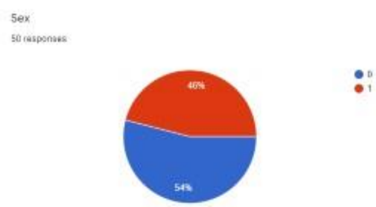


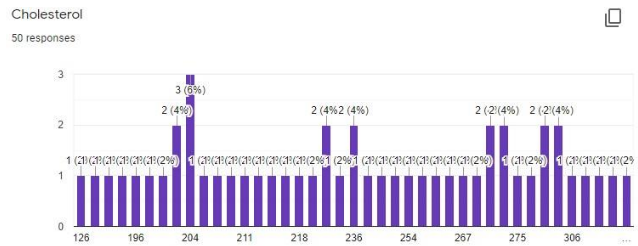
Google Form for collecting unstructured data

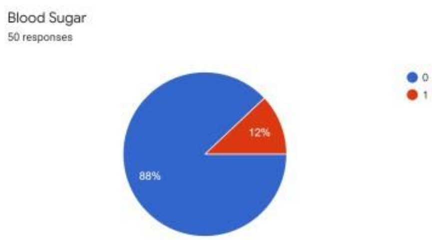


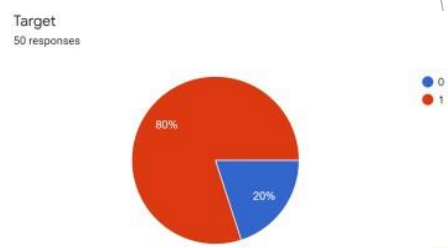
Data Analysis on Unstructured Data







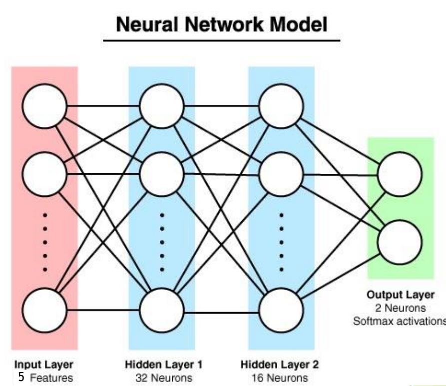




Neural Network by Harsh

```python
# Import packages that we will be working with.
import os
import numpy as np
import pandas as pd
from keras.layers import Dense
import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow.python.keras.models import Sequential
from tensorflow.python.keras.layers import Dense
from keras.optimizers import Adam
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, accuracy_score
```

```python
# Load the dataset, and view couple of the first rows.
data = pd.read_csv("heart1.csv")
print(data.head(3))

# Check the datatypes
print(data.dtypes)
```




```
   age  sex  trestbps  chol  fbs  target
0   69    1       160   234    1       1
1   45    0       138   236    0       1
2   50    0       120   244    0       1
age         int64
sex         int64
trestbps    int64
chol        int64
fbs         int64
target      int64
dtype: object
```

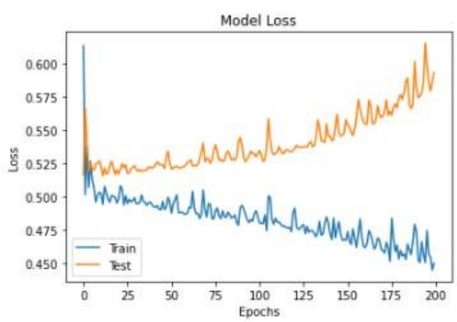
```python
Y = data.target.values
X = data.drop(['target'], axis=1)

# Now split to train/test with 80% training data, and 20% test data.
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_state=42)

# Check dimensions of both sets.
print("Train Features Size:", X_train.shape)
print("Test Features Size:", X_test.shape)
print("Train Labels Size:", Y_train.shape)
print("Test Labels Size:", Y_test.shape)
```

```
Train Features Size: (40, 5)
Test Features Size: (10, 5)
Train Labels Size: (40,)
Test Labels Size: (10,)
```

```python
history = model.fit(X_train, Y_train, validation_data=(X_test, Y_test), epochs=200, batch_size=16, verbose=2)
```

```
Epoch 1/200
3/3 - 3s - loss: 0.6136 - accuracy: 0.6500 - val_loss: 0.5165 - val_accuracy: 0.8000
Epoch 2/200
3/3 - 0s - loss: 0.5017 - accuracy: 0.8000 - val_loss: 0.5665 - val_accuracy: 0.8000
Epoch 3/200
3/3 - 0s - loss: 0.5386 - accuracy: 0.8000 - val_loss: 0.5418 - val_accuracy: 0.8000
Epoch 4/200
3/3 - 0s - loss: 0.5072 - accuracy: 0.8000 - val_loss: 0.5195 - val_accuracy: 0.8000
Epoch 5/200
3/3 - 0s - loss: 0.5269 - accuracy: 0.8000 - val_loss: 0.5251 - val_accuracy: 0.8000
Epoch 6/200
3/3 - 0s - loss: 0.5130 - accuracy: 0.8000 - val_loss: 0.5211 - val_accuracy: 0.8000
Epoch 7/200
3/3 - 0s - loss: 0.5067 - accuracy: 0.8000 - val_loss: 0.5200 - val_accuracy: 0.8000
Epoch 8/200
3/3 - 0s - loss: 0.4962 - accuracy: 0.8000 - val_loss: 0.5248 - val_accuracy: 0.8000
Epoch 9/200
3/3 - 0s - loss: 0.5010 - accuracy: 0.8000 - val_loss: 0.5255 - val_accuracy: 0.8000
Epoch 10/200
3/3 - 0s - loss: 0.5035 - accuracy: 0.8000 - val_loss: 0.5271 - val_accuracy: 0.8000
Epoch 11/200
3/3 - 0s - loss: 0.5025 - accuracy: 0.8000 - val_loss: 0.5231 - val_accuracy: 0.8000
Epoch 12/200
3/3 - 0s - loss: 0.4944 - accuracy: 0.8000 - val_loss: 0.5160 - val_accuracy: 0.8000
Epoch 13/200
3/3 - 0s - loss: 0.5079 - accuracy: 0.8000 - val_loss: 0.5217 - val_accuracy: 0.8000
Epoch 14/200
3/3 - 0s - loss: 0.5031 - accuracy: 0.8000 - val_loss: 0.5171 - val_accuracy: 0.8000
Epoch 15/200
3/3 - 0s - loss: 0.4983 - accuracy: 0.8000 - val_loss: 0.5180 - val_accuracy: 0.8000
Epoch 16/200
3/3 - 0s - loss: 0.4961 - accuracy: 0.8000 - val_loss: 0.5236 - val_accuracy: 0.8000
Epoch 17/200
3/3 - 0s - loss: 0.5012 - accuracy: 0.8000 - val_loss: 0.5269 - val_accuracy: 0.8000
Epoch 18/200
```

```python
# Plot the Loss function vs. number of Epochs
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Model Loss')
plt.ylabel('Loss')
plt.xlabel('Epochs')
plt.legend(['Train', 'Test'])
plt.show()
```




```python
predictions = np.argmax(model.predict(X_test), axis=1)
model_accuracy = accuracy_score(Y_test, predictions)*100
print("Model Accracy:", model_accuracy,"%")
print(classification_report(Y_test, predictions))
```

```
Model Accracy: 80.0 %
              precision    recall  f1-score   support

           0       0.00      0.00      0.00         2
           1       0.80      1.00      0.89         8

    accuracy                           0.80        10
   macro avg       0.40      0.50      0.44        10
weighted avg       0.64      0.80      0.71        10
```

Comparison of all algorithms

- We have covered normal classification algorithms like the Random forest and the XGBoost and also neural network algorithms.
- Since the data for this problem was normal numerical data the neural networks didn't have a clear cut advantage which they usually have in case of images.
- Thus they performed almost the same on structured data.

Comparison (contd…)

	Accuracy
Random Forest	89%
XGBoost	88.52%
Neural network on structured data	88.50%