

PROJECT SYNOPSIS ON: Heart Attack Prediction
Midterm Report Submitted in Partial Fulfillment of the
Requirements
for the degree of
Bachelor of Technology in Applied Electronics and
Instrumentation Engineering

By

Md Irshad

Nishant Shrivastava

Mithlesh Yadav

Satyam Anand

Vishnu Kumar

13005317046

13005317042

13005317045

13005317026

13005317006

Under the guidance of
Subhajit Bhowmick
Techno Main, Salt Lake
EM 4/ Salt lake City, Sector V
Kolkata – 700091
2021

Abstract:

Our main focus in this project is to build an IoT device that can predict a person's risk of heart attack. Here we have used a Machine Learning Algorithm (Support Vector Machine) which is a classification algorithm for prediction. Basically the heart of our project is Python Machine learning model and using Web based programming we will perform data manipulation from the cloud and serve it in the client side using latest technologies and the hardware part will be running on the WiFi Based Arduino setup with a sensor which measures the heart rate of the person.

By performing Data Analysis and Data Analytics we have found some crucial variable for our effective prediction

These variables are very important variables which will be user inputs. They will be fed with our IoT sensor data and then our model will predict.

In total for the purpose of classification we have chosen 5 variables.

1. Sex (enter by user)
2. Age (enter by user)
3. Cholesterol (Programmatically added data)
4. Blood pressure (Programmatically added data)
5. Heart Rate (Sensor data)

Due to some limitations like access to more hardware and software packages we will not be able to measure the values of Cholesterol and Blood pressure that's why we will be trying a programmatic way to solve this problem.

We will be hard coding some approx values according to the age of users.

We choose SVM classifier as our preferred algorithms because our dataset is small . If we get some more data then we will be using deep neural networks.

There are three layer of this project

1. Python ML Model (SVM classifier)
2. IoT device
3. Connecting Server and Client. (Example: Web Application)

Table of contents/Index:

| | |
|--|----|
| ● Introduction | 4 |
| ● Detailing of the project work and simulation | 5 |
| ● Result Set | 8 |
| ● Discussion | 12 |
| ● Remaining Work | 13 |
| ● References and Bibliography | 14 |

Introduction:

Heart diseases are the main reasons for death worldwide. According to the survey of the World Health Organization (WHO), 17.5 million total global deaths occur because of heart attacks and strokes. More than 75% of deaths from cardiovascular diseases (CVD) occur mostly in middle-income and low-income countries.

Also, 80% of the deaths that occur due to CVDs are because of stroke and heart attack. Therefore, detection of cardiac abnormalities at the early stage and tools for the prediction of heart diseases can save a lot of life and help doctors to design an effective treatment plan which ultimately reduces the mortality rate due to cardiovascular diseases. Due to the development of advanced healthcare systems, lots of patient data are nowadays available at open source which can be used for designing predictive models for Cardiovascular diseases. Machine learning is a discovery method for analyzing big data from an assorted perspective and encapsulating it into useful information.

Nowadays, a huge amount of data pertaining to disease diagnosis, patients etc. are generated by healthcare industries. Data mining provides a number of techniques which discover hidden patterns or similarities from data. Therefore, in this paper, a machine learning algorithm is proposed for the implementation of a heart disease prediction system which was validated on two open access heart disease prediction datasets.

The sensors collect the data after a specific time, analyze it and use it to initiate the required action, and provide an intelligent cloud-based network for analysis, planning and decision making.

Detailing of the project work and simulation:

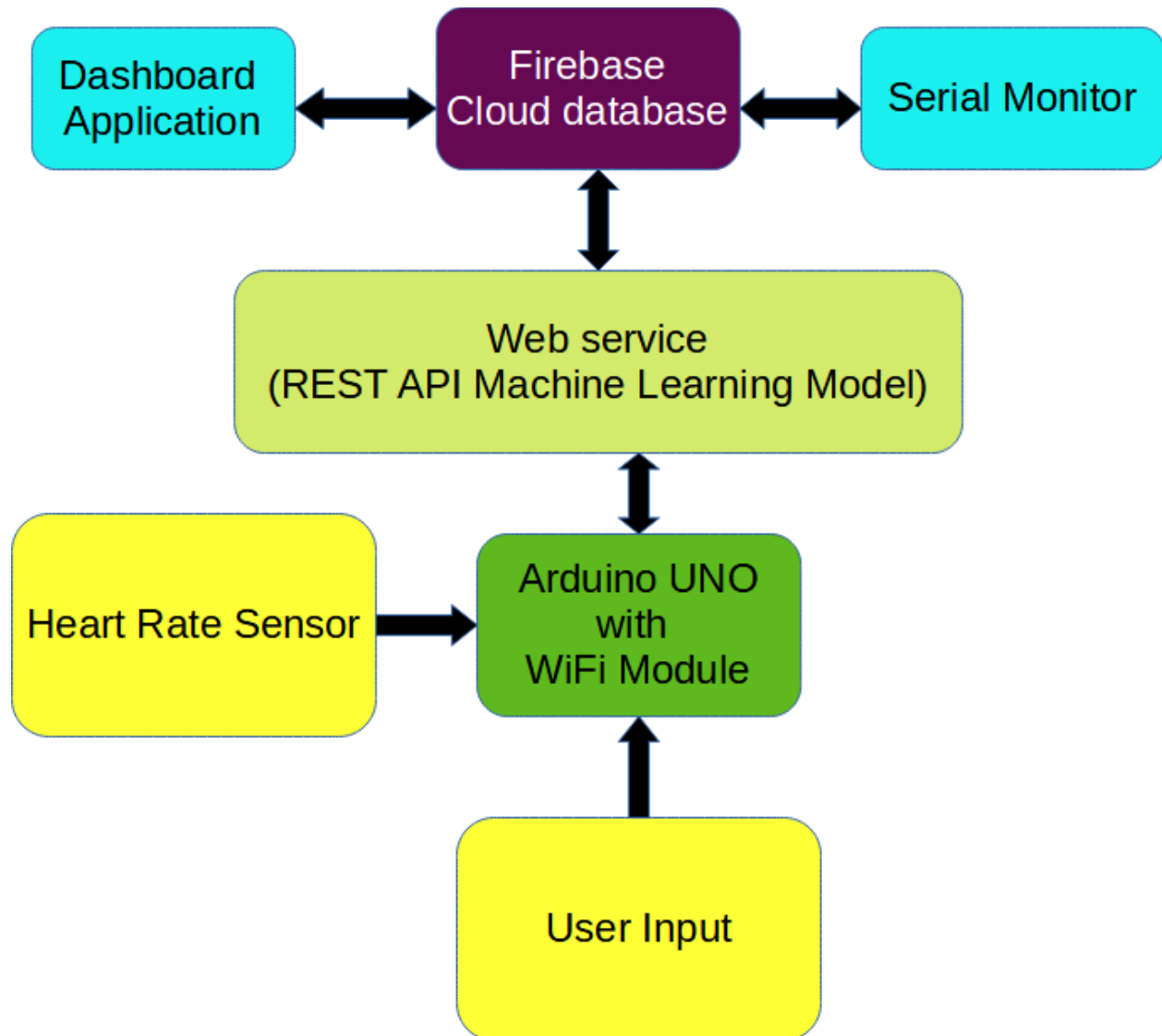


Fig 1. Block Diagram

With the provided user inputs to the arduino device the data will be send to a Web service(REST API) as a POST request which will save the predicted output by the Machine learning model to firebase cloud database. The returned data will be shown to the serial monitor of the Arduino device and in the dashboard application which is our Web Application.

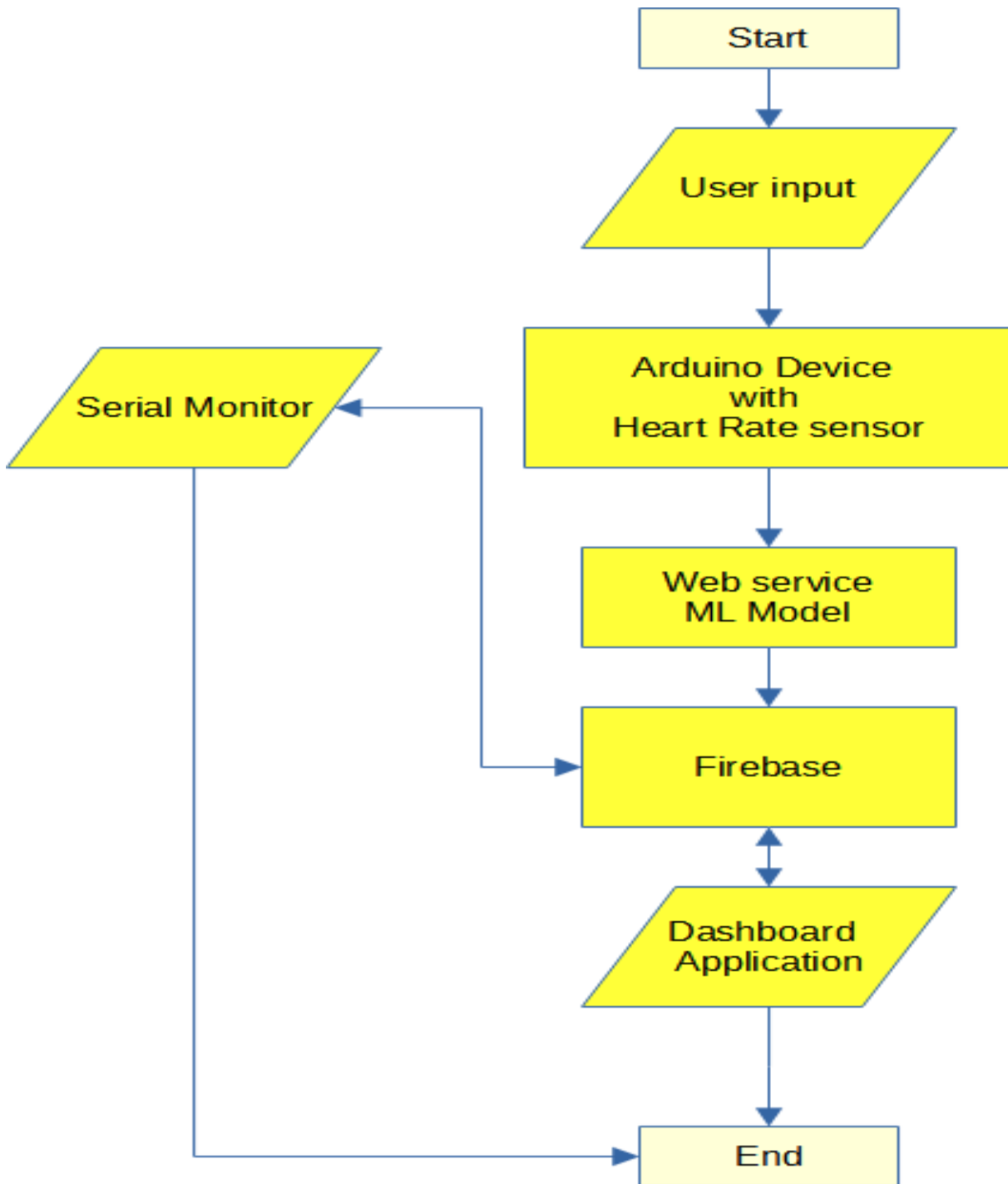


Fig 2. Flow Chart

This Fig 2. is the flow chart of our project. After every iteration of data being saved to the database a success message will be show up in the serial monitor of our arduino device.

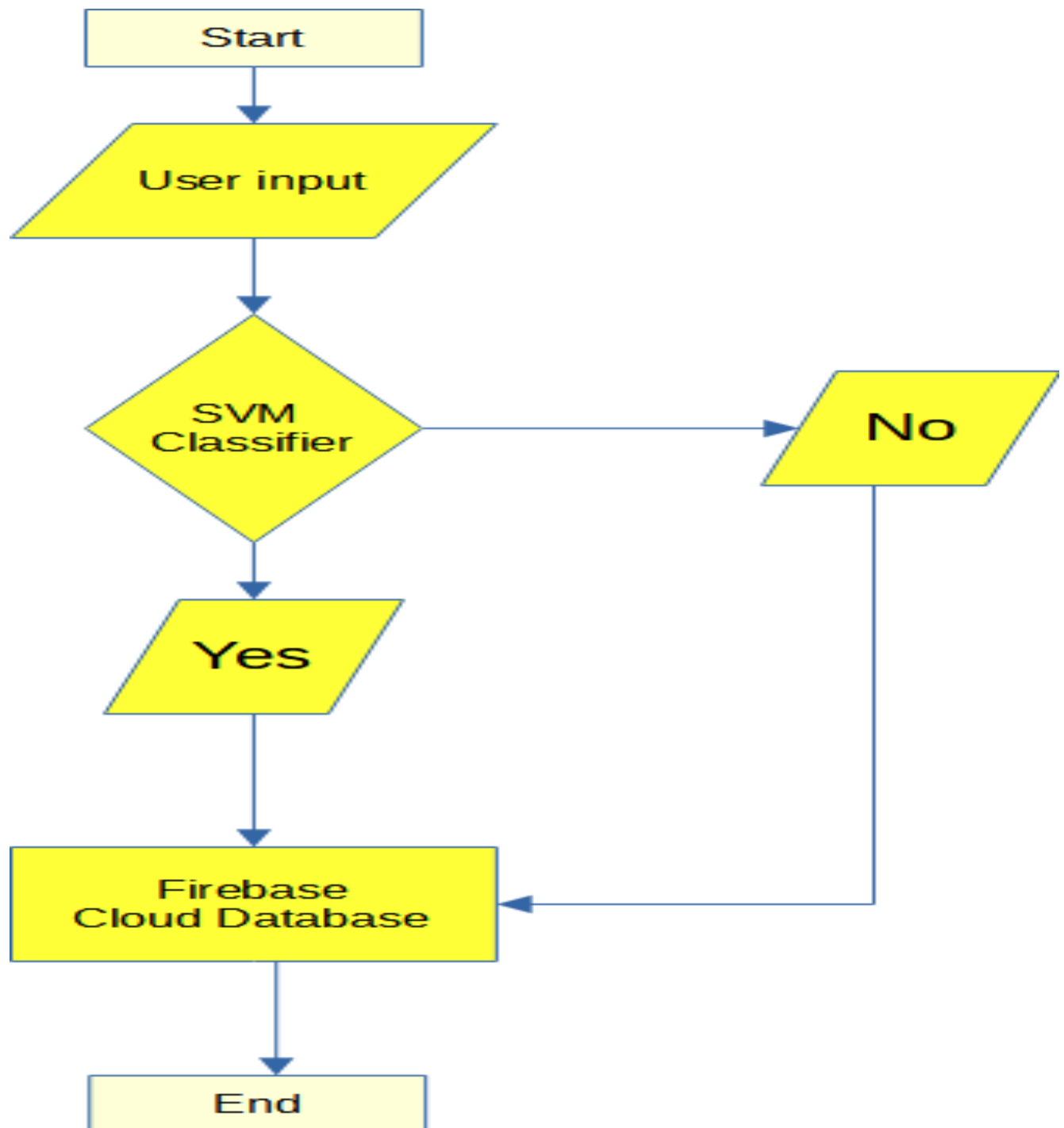
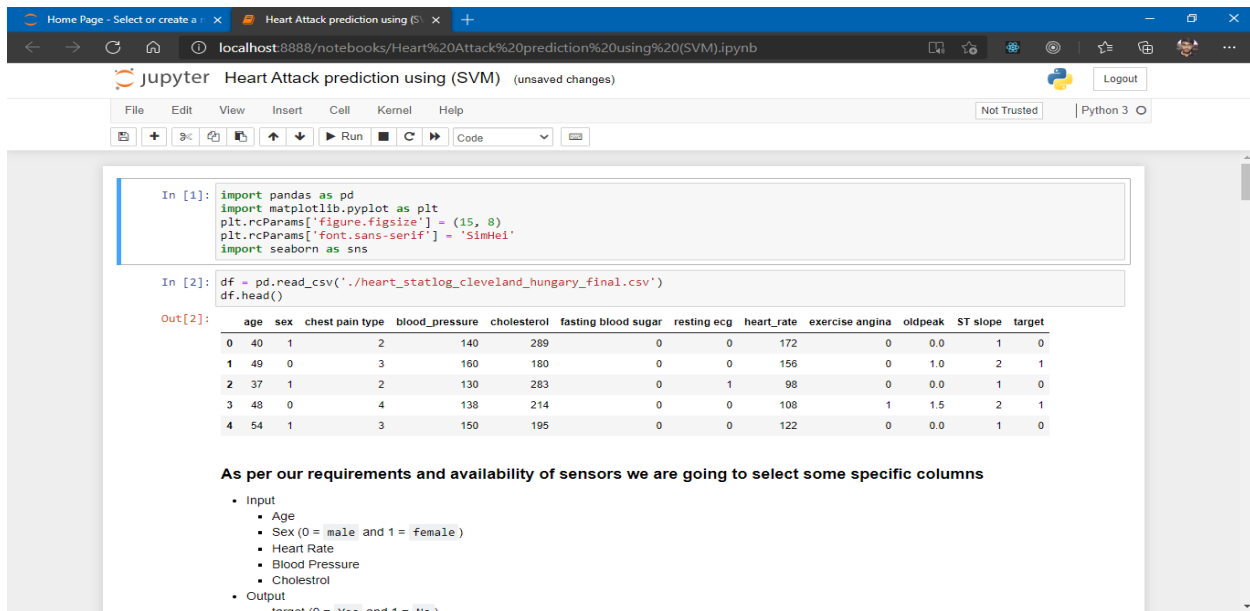


Fig 3. Machine Learning Model Flow chart

Using this Fig 3. We can see the flow of our Classifier model.

Result Set:

In the first step we gathered data from online resource to build our model to predict



The screenshot shows a Jupyter Notebook titled "Heart Attack prediction using (SVM)". The first cell imports necessary libraries: pandas, matplotlib.pyplot, and seaborn. The second cell reads a CSV file named "heart_statlog_cleveland_hungary_final.csv" and displays the first five rows of the dataset. The output shows a table with columns: age, sex, chest pain type, blood_pressure, cholesterol, fasting blood sugar, resting ecg, heart_rate, exercise angina, oldpeak, ST slope, and target. Below the table, a text block states: "As per our requirements and availability of sensors we are going to select some specific columns". A bulleted list follows, categorizing columns into "Input" (Age, Sex, Heart Rate, Blood Pressure, Cholesterol) and "Output" (target).

```
In [1]: import pandas as pd
import matplotlib.pyplot as plt
plt.rcParams['figure.figsize'] = (15, 8)
plt.rcParams['font.sans-serif'] = 'SimHei'
import seaborn as sns

In [2]: df = pd.read_csv('./heart_statlog_cleveland_hungary_final.csv')
df.head()

Out[2]:
```

| | age | sex | chest pain type | blood_pressure | cholesterol | fasting blood sugar | resting ecg | heart_rate | exercise angina | oldpeak | ST slope | target |
|---|-----|-----|-----------------|----------------|-------------|---------------------|-------------|------------|-----------------|---------|----------|--------|
| 0 | 40 | 1 | 2 | 140 | 289 | 0 | 0 | 172 | 0 | 0.0 | 1 | 0 |
| 1 | 49 | 0 | 3 | 160 | 180 | 0 | 0 | 156 | 0 | 1.0 | 2 | 1 |
| 2 | 37 | 1 | 2 | 130 | 283 | 0 | 1 | 98 | 0 | 0.0 | 1 | 0 |
| 3 | 48 | 0 | 4 | 138 | 214 | 0 | 0 | 108 | 1 | 1.5 | 2 | 1 |
| 4 | 54 | 1 | 3 | 150 | 195 | 0 | 0 | 122 | 0 | 0.0 | 1 | 0 |

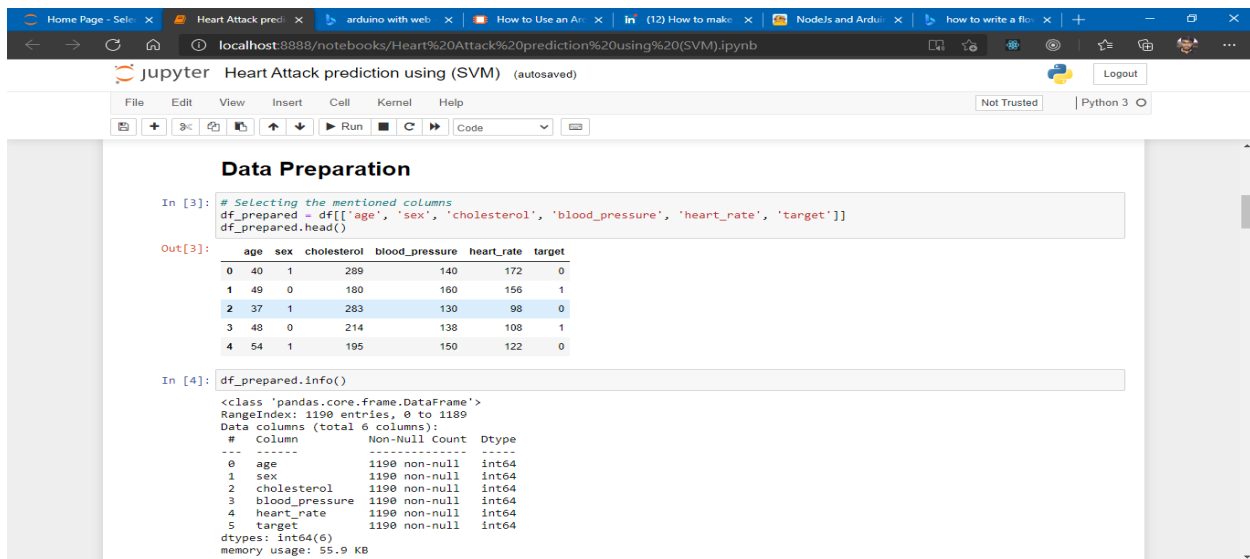
As per our requirements and availability of sensors we are going to select some specific columns

- Input
 - Age
 - Sex (0 = male and 1 = female)
 - Heart Rate
 - Blood Pressure
 - Cholesterol
- Output
 - target (0 = Yes and 1 = No)

Data From Kaggle.com

In the beginning we first eliminated some attributes which we cannot calculate.

After reducing the datasets it became easy to analyze



The screenshot shows the same Jupyter Notebook with additional code for data preparation. The third cell selects specific columns (age, sex, cholesterol, blood_pressure, heart_rate, target) and displays the first five rows of the resulting dataframe. The fourth cell uses the info() method to provide summary statistics for the dataframe, showing 1190 entries and 6 columns.

```
In [3]: # Selecting the mentioned columns
df_prepared = df[['age', 'sex', 'cholesterol', 'blood_pressure', 'heart_rate', 'target']]
df_prepared.head()

Out[3]:
```

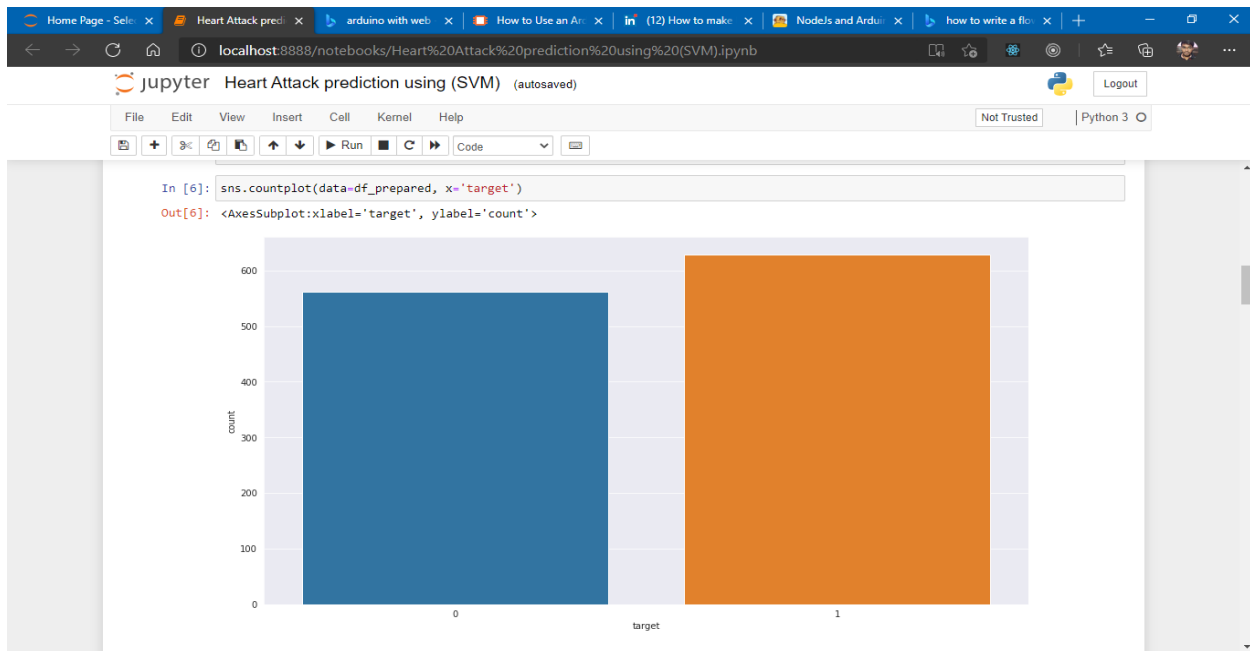
| | age | sex | cholesterol | blood_pressure | heart_rate | target |
|---|-----|-----|-------------|----------------|------------|--------|
| 0 | 40 | 1 | 289 | 140 | 172 | 0 |
| 1 | 49 | 0 | 180 | 160 | 156 | 1 |
| 2 | 37 | 1 | 283 | 130 | 98 | 0 |
| 3 | 48 | 0 | 214 | 138 | 108 | 1 |
| 4 | 54 | 1 | 195 | 150 | 122 | 0 |

```
In [4]: df_prepared.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1190 entries, 0 to 1189
Data columns (total 6 columns):
#   Column          Non-Null Count  Dtype
---  -
0   age             1190 non-null   int64
1   sex             1190 non-null   int64
2   cholesterol      1190 non-null   int64
3   blood_pressure   1190 non-null   int64
4   heart_rate       1190 non-null   int64
5   target          1190 non-null   int64
dtypes: int64(6)
memory usage: 55.9 KB
```

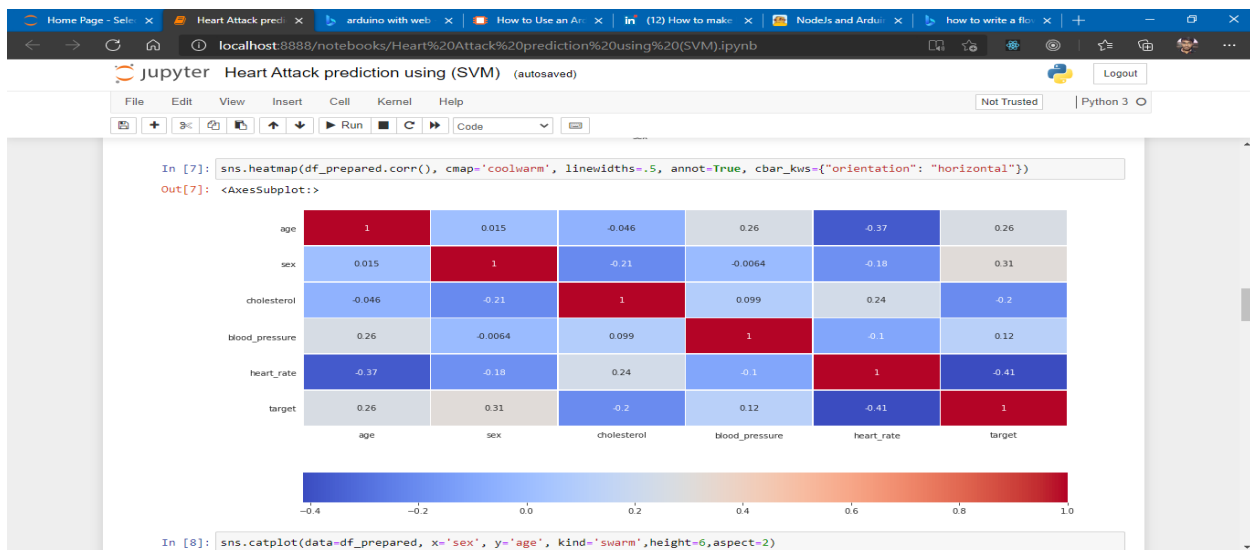
Data Preparation

There are no missing data points and we have chosen some data points which are relevant to us.



Data Analysis

By Performing analysis we found that data is not much gender biased it's distribution is quite acceptable for prediction



Visualizing Correlations

Heat-map calculated by taking Pearson's coefficient as [-1, 1]

Using these types of graphs we can say which attribute is correlated more or less with another attribute or is there any particular relationship. **For example** blood pressure has a little amount of impact in heart attack.

```

In [17]: # We need to scale the values for better accuracy in prediction
from sklearn.metrics import classification_report, auc, confusion_matrix, plot_roc_curve
from sklearn.preprocessing import MinMaxScaler

In [18]: scaler = MinMaxScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.fit_transform(X_test)

In [19]: model_svc = SVC(probability=True)
model_svc.fit(X_train, y_train)

Out[19]: SVC(probability=True)

In [20]: y_pred = model_svc.predict(X_test)

In [21]: Output = pd.DataFrame({
    'Test Data': y_test,
    'Predicted Data': y_pred,
    'Error': -(y_test-y_pred)})

In [22]: Output['Error'].value_counts()

Out[22]:
Error
0    139
1     26
-1    14
Name: Error, dtype: int64

In [23]: print(classification_report(y_test, y_pred, target_names=['Yes', 'No']))

precision    recall  f1-score   support

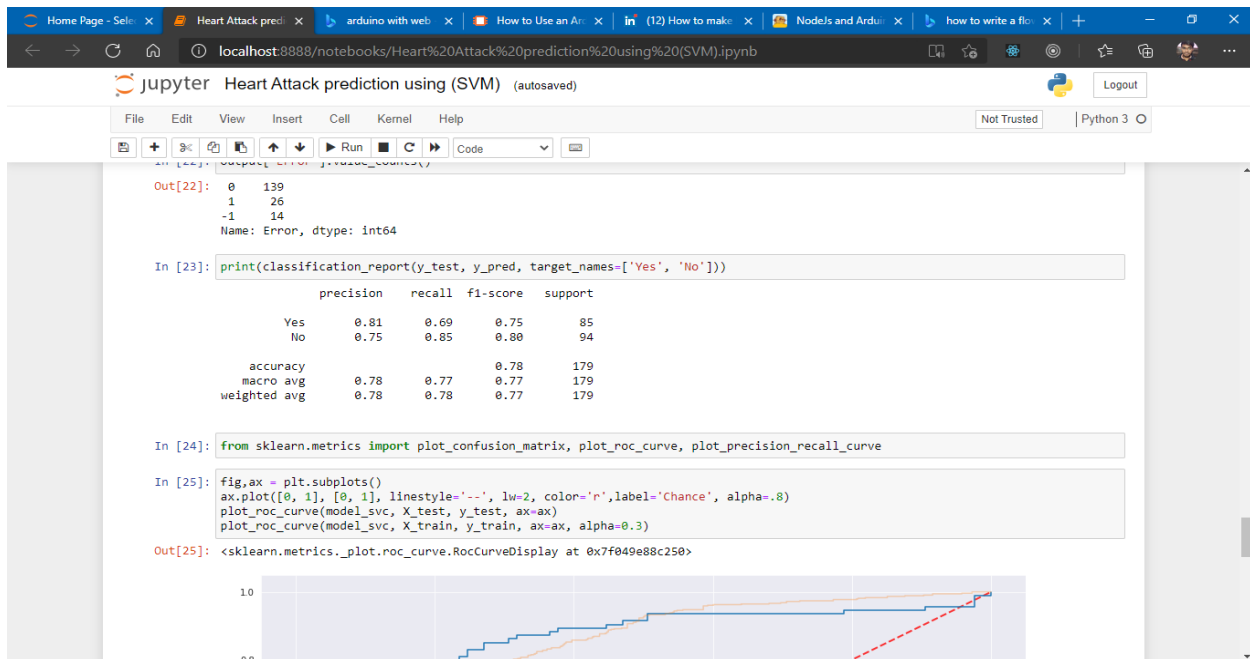
Yes         0.81     0.69     0.75         85
No          0.75     0.85     0.80         94

accuracy          0.78
macro avg         0.78     0.77     0.77         179
weighted avg      0.78     0.78     0.77         179

```

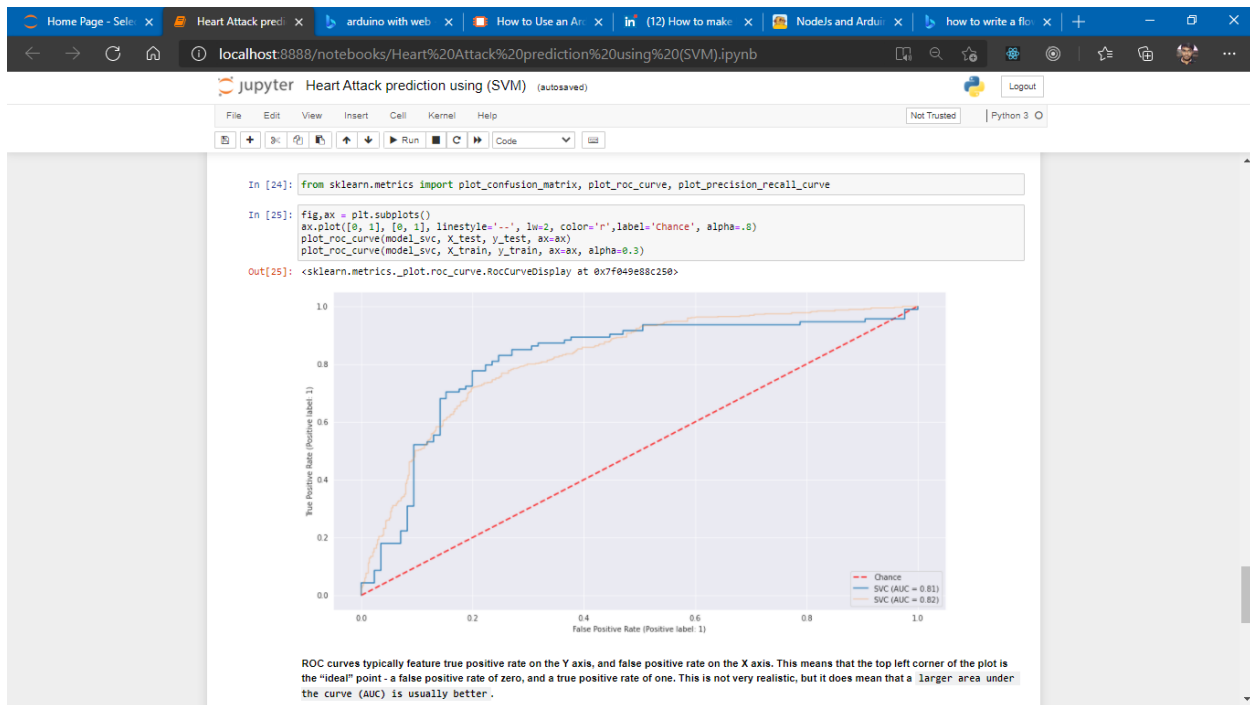
Choosing ML model Support Vector Machines (SVM)

In order to work with data points we need to scale those points within the range of -1 and 1. For the ease in calculation and correct prediction



Classification Report

1. We can see that precision of predicting the attribute Yes is more than that of No. And the accuracy of our model is also acceptable as we have less data right now.



ROC-AUC Curve

Our training and testing data are somewhat working the same as we are expecting from our calculation.

```
what is CS? x EJS vs Puig x Quick Start x Schema bu x GraphQL x Preference x bitbucket x Home Page x Heart Atta x + - x
localhost:8888/notebooks/Heart%20Attack%20ML%20Prediction%20(Model).ipynb
jupyter Heart Attack ML Prediction (Model) (unsaved changes)
File Edit View Insert Cell Kernel Help Not Trusted Python 3
In [1]: import pickle
In [2]: model = pickle.load(open('./model_svc.pkl', 'rb'))

A function to test our model
In [3]: def UsingModel(X_data):
...
...
...
X_data: It is 2 Dimensional array of various inputs
Our function returns nothing but we want it to print the prediction.
...
from sklearn.preprocessing import MinMaxScaler
minmax = MinMaxScaler()
X_data = minmax.fit_transform(X_data)
output = model.predict(X_data)
if output == 0:
    print("You have a chance of Heart attack")
else:
    print("You have no chance of Heart attack")
return

In [4]: X_1 = [[25, 1, 289, 140, 145]]
UsingModel(X_1)
You have no chance of Heart attack
```

Testing out the model we created

Our model prediction accuracy is good enough as the data grows up model needs to be re-tuned.

Discussion:

In this project our goal is to predict the heart attack. Using Web Technologies, Machine learning and IoT concepts we have tried to achieve this goal.

We have three things in mind:-

1. Data is being entered by the user in the device.
2. Data will be send to the web server using REST API web services and the output will be saved to the database.
3. If a user wants he/she will be able to visualize their data because of dashboard functionality.

In the beginning we have a small amount of data so our work is not that complicated but as soon the data will grow we would require to upgrade a lot of things as everything will become more complex to handle like the performance of a model will decrease after every use. So In order to solve this problem I need to work on automated pipelines for data processing to training and testing of models. In the meantime we also need to work on scaling some aspects of the server to optimize the load of many users.

There are more attributes we can take to build a sophisticated model for prediction but because of limitation in resources both financially and educationally. We cannot select them for example there is an attribute called cholesterol which cannot be measured by some simple sensor for that we need some more advanced tools. Which are out of reach for now.

Well in terms of hardware we can also choose from lots of resources like Raspberry Pi which is good for our project but the fact is that because of less accessibility we are using Arduino with a WiFi board. Specially the sensor we are using might not be well calibrated which is very important if the provided data is garbage then our prediction is useless and hence this project will become worthless.

In the whole process of sending and receiving data there is a huge amount of network calls which our server and client needs to perform to send and receive data which will cause some latency and more data will be consumed. We can avoid this problem by using web sockets.

If this project scales up then I will consider all the points and try to find an optimal solution to both hardware and software resources.

Remaining work:

- Hardware Development
- Arduino program
- Web Application (Frontend and Backend)
- Deployment

References and Bibliography:

Sample References:

1. M. Sanjeev Arulampalam, Simon Maskell, Neil Gordon, and Tim Clapp, “A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking”, IEEE Transactions on Signal Processing, Vol. 50, No. 2, pp:174-188, Feb’ 2002.
2. R. N. Banavar, J. L. Speyer, “Properties of Risk-Sensitive Filters/Estimators”, IEEE Proceedings of Control Theory Application, Vol.145, No. 1, January 1998.
3. R. G. Brown, and P. Y. C. Hwang, Introduction to Random Signals and Applied Kalman Filtering with Matlab Exercises and Solutions, 3rd Edition, John Wiley & Sons, Inc, 1997.
4. Universal Description, Discovery and Integration, UDDI ; <http://www.uddi.org>; October 5, 2007.

Appendix

Code

```
# We need to scale the values for better accuracy in prediction

from sklearn.metrics import classification_report, auc, confusion_matrix, plot_roc_curve

from sklearn.preprocessing import MinMaxScaler

Scaler = MinMaxScaler()

X_train = Scaler.fit_transform(X_train)

X_test = Scaler.fit_transform(X_test)

model_svc = SVC(probability=True)

model_svc.fit(X_train, y_train)

y_pred = model_svc.predict(X_test)

print(classification_report(y_test, y_pred, target_names=['Yes', 'No']))

from sklearn.metrics import plot_confusion_matrix, plot_roc_curve, plot_precision_recall_curve

fig,ax = plt.subplots()

ax.plot([0, 1], [0, 1], linestyle='--', lw=2, color='r',label='Chance', alpha=.8)

plot_roc_curve(model_svc, X_test, y_test, ax=ax)

plot_roc_curve(model_svc, X_train, y_train, ax=ax, alpha=0.3)

plot_confusion_matrix(model_svc, X_test, y_test)
```

```
import pickle

# Saving the model

filename='model_svc.pkl'

pickle.dump(model_svc, open(filename, 'wb'))

df_prepared.to_csv('prepared_data.csv', index=False)
```