Artificial Intelligence (CSE3013)

Final Report

Heart Disease Prediction Application

Mrinalini Singh(18BCE1001)

Ishita Saluja (18BCE1100)

Faculty: Priyadarshini. J

School of Computer Science and Engineering

Vellore Institute of Technology, Chennai

Acknowledgement

We wish to express our sincere thanks and deep sense of gratitude to our project guide Priyadarshini J, SCOPE, 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

1 Abstract and Introduction

1.1 Abstract

Heart disease, alternatively known as cardiovascular disease, encases various conditions that impact the heart and is the primary basis of death worldwide over the span of the past few decades. It associates many risk factors in heart disease and a need of the time to get accurate, reliable, and sensible approaches to make an early diagnosis to achieve prompt management of the disease. Heart disease classification based on electrocardiogram (ECG) signal has become a priority topic in the diagnosis of heart diseases because it can be obtained with a simple diagnostic tool of low cost. Since early detection of heart disease can enable us to ease the treatment as well as save people's lives, accurate detection of heart disease using ECG is very important.

So, in this project we created a user-friendly application which deploys two machine learning models. One takes user info and predicts the criticality of heart disease by random forest and the other takes ECG signal files and pre-processes them using wfdb and use CNN to predict the type of arrythmia. The website was made using HTML/CSS and the server using python.

1.2 Introduction

The early detection of abnormal heart conditions is vital to identify heart problems and avoid sudden cardiac death. The people with similar heart conditions almost have similar electrocardiogram (ECG) signals. By analysing the ECG signals' patterns one can predict arrhythmias. Since the conventional methods of arrhythmia detection rely on observing morphological features of the ECG signals which are tedious and very time consuming, the automatic detection of arrhythmia is more preferable. In order to automate detection of heart diseases an adequate algorithm is required which could classify the ECG signals with unknown features according to the similarities between them and the ECG signals with known features. If this classifier can find the similarities precisely, the probability of arrhythmia detection is increased and this algorithm can become a useful means in laboratories.

Since in laboratories, arrhythmias are detected by continuous monitoring of the ECG signals which is very time consuming, this project presents a new method in detecting heart diseases in an automatic manner. In automatic arrhythmia detection, the accuracy is the most important factor. Here, two classifiers are presented to increase the accuracy based on the fact that people with similar heart condition almost have similar ECG signals, so the parameters of the ECG model for them are almost identical. In this method the status of a new ECG signal is predicted through estimating its parameters and finding the similarities between them and the parameters of the ECG signals with known heart conditions.

The electrocardiogram (ECG) signal reflects the electrical activity of the heart observed from the strategic points of the human body and represented by quasi-periodic voltage signal. The ECG signal contains essential information about the cardiac pathologies affecting the heart, characterized by five peaks known as fiducial points, which are represented by the letters P, Q, R, S, and T. The QRS complex is the depolarization of the right and left heart ventricles, which is used as a reference point for signal analysis. The P wave is the result of the depolarization of the atrium, while the ventricle causes the rest of the peaks. The diagnosis of the signal relies on the morphology of the waves, as well as the duration of each peak and the segments that make it up. Therefore, detection of each section of the ECG signal is essential for health professionals in screening, diagnosis, and monitoring of several heart conditions.

Our aim was to create a user-friendly application which deploys two machine learning models. First takes user information and predicts the criticality of heart disease and the second takes ECG signal files pre-processes them, find QRS and then predicts the kind of arrythmia.

2 Literature Survey

ECG signals are considered as best among any other measurement to identify the heart disease. Some of the literature proven that the need of ECG and machine learning algorithms which helpful for classifying the types of heart diseases.

In this section, some of the research articles are discussed to show the importance of ECG and machine learning algorithms. In addition, it also provides the way of approaching the signals with further enhancement needed in the existing system.

2.1 Identification and Classification of Heart Beat by Analysing ECG Signal using Naive Bayes [1]

2.1.1 **Paper methodology:**

In this paper the arrythmia prediction was carried out by first finding the time interval, between the individual heart was one of the features and one wave segment was mapped with another wave segment to extract the dissimilarity between the frequency of continuous segments and then Naïve Bayes classifier was used to classify the signals.

2.1.2 Method advantage:

This paper proposed a new feature extraction concept to improve the classification accuracy of predicting the heart diseases. Naive Bayes is used for classifying the normal signals and abnormal based on the values given by feature which is extracted from the given ECG signals. And also, results show that the proposed system provides better accuracy due to its feature extraction concept and classify the heart signals with high probability.

2.1.3 Method disadvantage:

Here, the heart signals are classified as normal or abnormal only. It does not take into account the various types of diseases and classify them.

2.1.4 Comparison with our project:

Our project QRS segmentation for feature extraction which performed better than time difference approach and this paper just predicts if the signal is normal or abnormal it doesn’t give the type of the disease which is covered in our project.

2.2 Classification of Arrhythmia Using Machine Learning Techniques [2]

2.2.1 Paper methodology:

In the paper, the original data set was partitioned into two mutually disjoint sets: a training set and a test set. The training set was used to train the learning algorithm, and the induced decision rules were tested on the test set. The settings used in the experiments were as follows. The OneR, J48 and Naïve

Bayes were used. The cross validation was set to 10 and all other settings were the WEKA program defaults.

2.2.2 Method advantage:

The highest accuracy was observed in the case of decision-tree induction algorithm (J48) compared with the training data itself. Despite the high accuracy rate of J48, the accuracy curve is unstable when the data is spilt into training and test, whereas OneR and Naïve Bayes show stable accuracy for the same dataset. The accuracy rate of OneR is the lowest among the three algorithms.

2.2.3 Method disadvantage:

This method doesn’t use ECG signals directly and just uses numerical database. This paper also does not use feature extraction and directly applies machine learning algorithm so it cannot be used in real time.

2.2.4 Comparison with our project:

Our project uses raw signal data to predict heart disease. The raw signal is pre-processed which is better since it provides for better real time application

2.3 Heart diseases prediction based on ECG signals’ classification using a genetic-fuzzy system and dynamical model of ECG signals [3]

2.3.1 Paper methodology:

In this paper arrhythmias prediction is carried out by analysing ECG signals with the help of genetic fuzzy system. Genetic fuzzy systems classify the signals as known and unknown. After that, it analysed the unknown signals that any similarities can identify based on the features given for heart diseases. Once it identified, fuzzy classifier classifies the type of heart disease based on the relation between the features segregated.

2.3.2 Method advantage:

The ECG signals are classified with accuracy of 93.34%. For increasing the accuracy of the classification, the membership functions and fuzzy rules are optimized through the GA leading to the introduction of a newly constructed genetic-fuzzy classifier. The classifier and its simulation results indicate an increase in the accuracy up to 98.67%

2.3.3 Method disadvantage:

The paper couldn’t reach as much accuracy as it could.

2.3.4 Comparison with our project:

The performance of our model was much better than the performance of their model.

2.4 Classification of ECG Arrhythmia using Recurrent Neural Networks [4]

2.4.1 Paper methodology:

This paper, used Recurrent Neural Networks to go ahead with the classification and diagnosis of the arrhythmic beats. The effectiveness of the heartbeat classification using RNN is carried out by the percentage of accuracy, specificity, and sensitivity. The model was implemented directly using the signals from the MIT BIH database and no pre-processing had been done.

2.4.2 Method advantage:

The model is implemented directly using the signals from the MIT BIH database and no pre-processing has been done. Therefore, the complexity of our implemented model is much lesser than traditional machine learning algorithms

2.4.3 Method disadvantage:

This paper directly uses signals without any pre-processing and just does binary classification and not multi-class classification and even then, the accuracy is not high.

2.4.4 Comparison with our project:

Our project uses signal processing library and pre-processes data which increases the accuracy a lot more along with a better model and our project also does multi class classification rather than just binary classification hence performs much better than the given paper overall.

2.5 Heart Disease Prediction using Machine Learning Techniques [5]

2.5.1 Paper methodology:

Aim of this research is to predict whether or not a patient will develop heart disease. This research was done on supervised machine learning classification techniques using Naïve Bayes, decision tree, random forest, and K-nearest neighbour on UCI repository. Various experiments using different classifier algorithms were conducted through the WEKA tool. Dataset was classified and split into a training set and a test set. Pre-processing of the data is done and supervised classification techniques such as Naïve Bayes, decision tree, K-nearest neighbour, and random forest are applied to get accuracy score. The accuracy score results of different classification techniques were noted using Python Programming for training and test data sets.

2.5.2 Method advantage:

This method reduces the number of attributes from 76 attributes to 14 attributes which decreases the processing time. This method also compares various machine learning models which helps find the best model.

2.5.3 Method disadvantage:

This method ones considers 4 machine learning models which doesn’t seem to be enough. This paper also doesn’t describe the method they have used for choosing the attributes and much more better techniques that can be used for feature extraction like PCA. This project also finds the best model as KNN with 7 neighbours which is very susceptible to overfitting.

2.5.4 Comparison with our project:

Our project compares more machine learning models and finds that random forest performs much better under various evaluation metrics than the model that they've chosen.

2.6 Heart Disease Prediction Using CNN Algorithm [6]

2.6.1 Paper Methodology:

We propose to use convolutional neural network algorithm as a disease risk prediction algorithm using structured and perhaps even on unstructured patient data. The accuracy obtained using the developed model ranges between 85 and 88%. We have proposed further by applying other machine learning algorithms over the training data to predict the risk of diseases, comparing their accuracies so that we can deduce the most accurate one.

2.6.2 Method advantage:

The major advantage of using CNN Algorithm compared to its predecessors is that it automatically detects the important features without any human supervision.

2.6.3 Method Disadvantage:

The accuracy obtained using the developed model ranges between 85 and 88% and the method is tedious.

2.6.4 Comparison with our project:

The accuracy of our project is much higher as compared to CNN Algorithm. We found that random forest performs much better under various evaluation metrics than the model that they've chosen.

2.7 Arrhythmia detection and classification using morphological and dynamic features of ECG signals [7]

2.7.1 Paper Methodology:

In this paper, a new approach for arrhythmia classification based on a combination of morphological and dynamic features is proposed. Wavelet Transform (WT) and Independent Component Analysis (ICA) are applied separately to each heartbeat to extract corresponding coefficients, which are categorized as 'morphological' features. In addition, RR interval information is also obtained characterizing the 'rhythm' around the corresponding heartbeat providing 'dynamic' features.

These two different types of features are concatenated and Support Vector Machine (SVM) is utilized for the classification of heartbeats into 15 classes.

2.7.2 Method Advantage:

The proposed method was tested over the entire MIT-BIH Arrhythmias Database [1] and it yields an overall accuracy of 99.66% on 85945 heartbeats, better than any other published results.

2.7.3 Method Disadvantage:

The major disadvantage of using ICA method is that there are limited number of electrodes (recordings) making it an overcomplete problem (non-square ICA). The other disadvantage is the distribution of BioSignal being close to Gaussian.

2.7.4 Comparison with our project:

The methodology applied in this paper is tedious as compared to the methods we have used in our project.

2.8 Arrhythmia Classification of ECG Signals Using Hybrid Features [8]

2.8.1 Paper Methodology:

In this paper, a novel method for classification of various types of arrhythmia using morphological and dynamic features is presented. Discrete wavelet transform (DWT) is applied on each heart beat to obtain the morphological features. It provides better time and frequency resolution of the electrocardiogram (ECG) signal, which helps in decoding important information of a quasiperiodic ECG using variable window sizes. RR interval information is used as a dynamic feature. The nonlinear dynamics of RR interval are captured using Teager energy operator, which improves the arrhythmia classification. Moreover, to remove redundancy, DWT sub bands are subjected to dimensionality reduction using independent component analysis, and a total of twelve coefficients are selected as morphological features. These hybrid features are combined and fed to a neural network to classify arrhythmia.

2.8.2 Method Advantage:

The proposed algorithm has been tested over MIT-BIH arrhythmia database using 13724 beats and MIT-BIH supraventricular arrhythmia database using 22151 beats. The proposed methodology resulted in an improved average accuracy of 99.75% and 99.84% for class- and subject-oriented scheme, respectively, using three-fold cross validation i.e. a very high accuracy.

2.8.3 Method Disadvantage:

Using discrete wavelet transform (DWT) for signal and image processing, it has three serious disadvantages: shift sensitivity, poor directionality, and lack of phase information. Moreover, Teager energy operator generates errors that can bias data and the excessive time needed to process a large number of trials.

2.8.4 Comparison with our project:

Application of random forest method proved to be much less tedious and much more efficient. The use of Teager energy operator generates errors hence decreasing the accuracy of the method.

2.9 Application of deep learning techniques for heartbeats detection using ECG signals-analysis and review [9]

2.9.1 Paper Methodology:

A five-class ECG dataset containing 100,022 beats is utilized for analysis of deep learning techniques. The constructed models were examined with this dataset, and results are presented. This study therefore provides information concerning deep learning approaches used for arrhythmia classification, and suggestions for further research in this area.

2.9.2 Method Advantage:

Various deep learning models and experimental studies are described and discussed rather than testing and analysing just one model.

2.9.3 Method Disadvantage:

The use of various deep learning techniques, makes the accuracy of this project much less. The training of the data set and examining each heartbeat takes up a lot of time with much less precision.

2.9.4 Comparison with our project:

Our project uses raw signal data to predict heart disease. The raw signal is pre-processed which is better since it provides for better real time application.

2.10 Classification of Heart Diseases Based On ECG Signals Using Long Short-Term Memory [10]

2.10.1 Paper Methodology:

In this paper, a classification method of heart diseases based on ECG by adopting a machine learning method, called Long Short-Term Memory (LSTM) is proposed, which is a state-of-the-art technique analysing time series sequences in deep learning.

2.10.2 Method Advantage:

The major advantage of using LSTM is that it not only achieves significantly better accuracy but also classifies heart diseases correctly in smaller response time than baseline techniques.

2.10.3 Method Disadvantage:

Prone to problems such as exploding and gradient vanishing. Due to its recurrent nature, the computation is slow. Training of RNN models can be difficult. If we are using relu or tanh as activation functions, it becomes very difficult to process sequences that are very long. Prone to problems such as exploding and gradient vanishing.

2.10.4 Comparison with our project:

Our project compares more machine learning models and finds that random forest performs much better under various evaluation metrics than the model that they’ve chosen.

3 Methodology

Project Link:

https://github.com/mrinalini1404/Heart_disease_prediction

https://colab.research.google.com/drive/1sQ6nI1bCw_i0pqxvte2x-llC3HWH0Lga?usp=sharing

3.1 Data Source

For this project, we have used dataset from UCI Machine learning repository and MIT-BIH Arrythmia database. The UCI data comprises a real dataset of 300 examples of data with 14 various attributes (13 predictors; 1 class) like blood pressure, type of chest pain, electrocardiogram result, etc. The MIT-BIH Arrhythmia Database contains 48 half-hour excerpts of two-channel ambulatory ECG recordings, obtained from 47 subjects studied by the BIH Arrhythmia Laboratory between 1975 and 1979. Twenty-three recordings were chosen at random from a set of 4000 24-hour ambulatory ECG recordings collected from a mixed population of inpatients (about 60%) and outpatients (about 40%) at Boston's Beth Israel Hospital; the remaining 25 recordings were selected from the same set to include less common but clinically significant arrhythmias that would not be well-represented in a small random sample. The recordings were digitized at 360 samples per second per channel with 11-bit resolution over a 10-mV range. Two or more cardiologists independently annotated each record; disagreements were resolved to obtain the computer-readable reference annotations for each beat (approximately 110,000 annotations in all) included with the database.

Links:

1. https://physionet.org/content/mitdb/1.0.0/
2. https://archive.ics.uci.edu/ml/datasets/heart+disease

3.2 Data Pre-processing

For MIT-BIH data as ECG beats are inputs of the proposed method, we suggest a simple and yet effective method for pre-processing ECG. Data was taken and processed using the wfdb python library. The steps used for extracting beats from an ECG signal are as follows:

1) Splitting the continuous ECG signal to 10s windows and select a 10s window from an ECG signal.

2) Normalizing the amplitude values to the range of between zero and one.

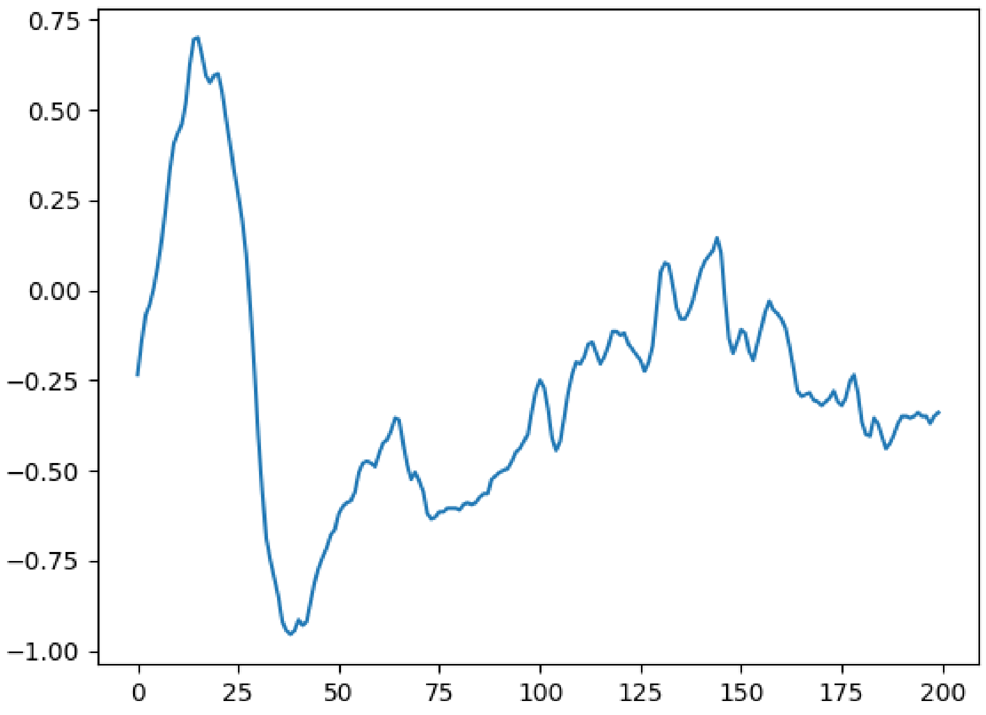
3) Finding the set of all local maximums based on zero crossings of the first derivative.

4) Finding the set of ECG R-peak candidates by applying a threshold of 0.9 on the normalized value of the local maximums.

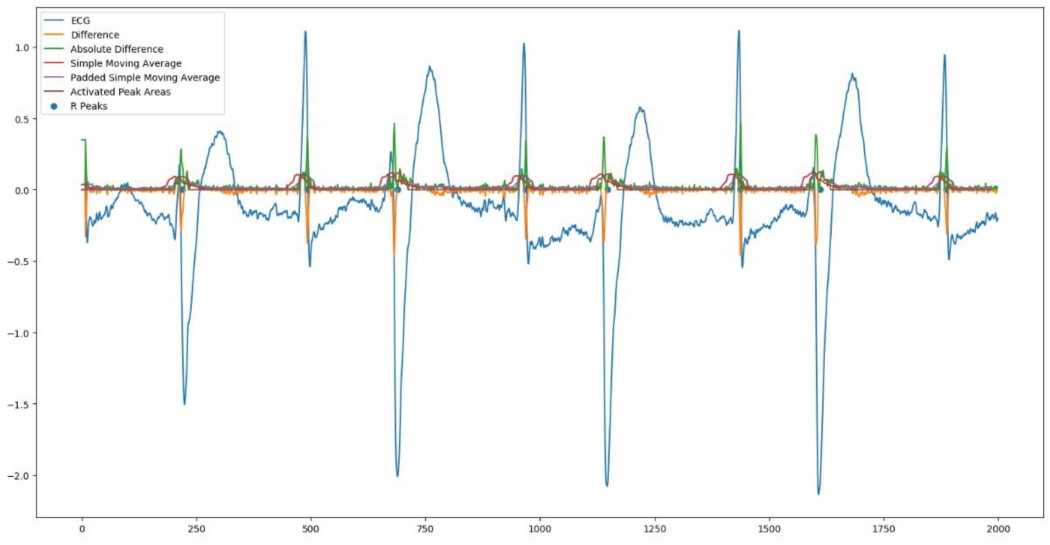
5) Finding the median of R-R time intervals as the nominal heartbeat period of that window (T).

6) For each R-peak, selecting a signal part with the length equal to 1.2T.

7) Padding each selected part with zeros to make its length equal to a predefined fixed length.



Extracted beat



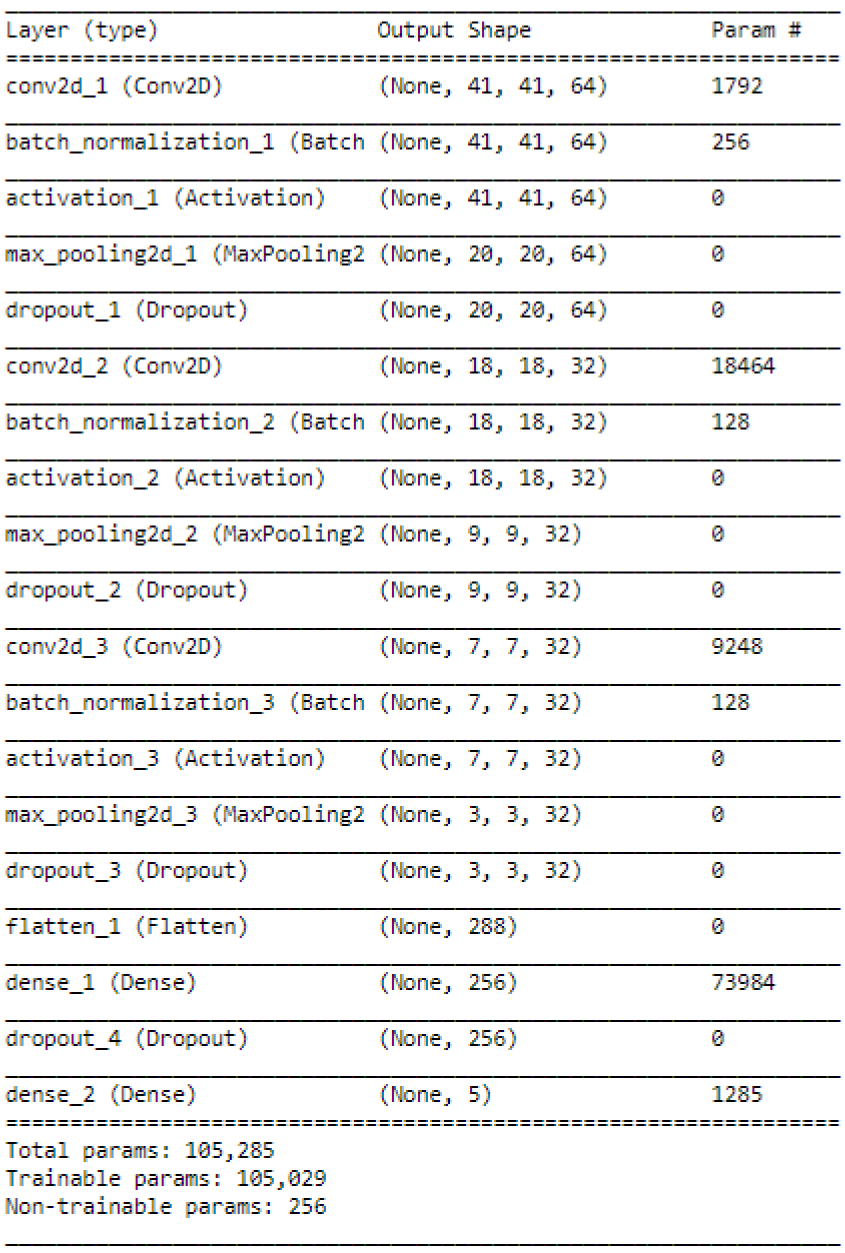
3.3 Training the classifier

3.3.1 For MIT-BIH dataset

In this project we are training a convolutional neural network for classification of ECG beat types on the MIT-BIH dataset. The trained network not only can be used for the purpose of beat classification, but also be used as an informative representation of heartbeats.

We compared various models and CNN turned out to be the best

```
cnn_model = Sequential ()

cnn_model.add(Conv2D(64, (3, 3), input_shape=input_shape))
cnn_model.add(BatchNormalization())
cnn_model.add(Activation('relu'))
cnn_model.add(MaxPooling2D(pool_size=(2, 2)))
cnn_model.add(Dropout(0.25))

cnn_model.add(Conv2D(32, (3, 3)))
cnn_model.add(BatchNormalization())
cnn_model.add(Activation('relu'))
cnn_model.add(MaxPooling2D(pool_size=(2, 2)))
cnn_model.add(Dropout(0.25))

cnn_model.add(Conv2D(32, (3, 3)))
cnn_model.add(BatchNormalization())
cnn_model.add(Activation('relu'))
cnn_model.add(MaxPooling2D(pool_size=(2, 2)))
cnn_model.add(Dropout(0.25))

cnn_model.add(Flatten())
cnn_model.add(Dense(256, activation='relu'))
cnn_model.add(Dropout(0.5))
cnn_model.add(Dense(5, activation='softmax'))
```



```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 41, 41, 64)        1792
_________________________________________________________________
batch_normalization_1 (Batch (None, 41, 41, 64)        256
_________________________________________________________________
activation_1 (Activation)    (None, 41, 41, 64)        0
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 20, 20, 64)        0
_________________________________________________________________
dropout_1 (Dropout)          (None, 20, 20, 64)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 18, 18, 32)        18464
_________________________________________________________________
batch_normalization_2 (Batch (None, 18, 18, 32)        128
_________________________________________________________________
activation_2 (Activation)    (None, 18, 18, 32)        0
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 9, 9, 32)          0
_________________________________________________________________
dropout_2 (Dropout)          (None, 9, 9, 32)          0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 7, 7, 32)          9248
_________________________________________________________________
batch_normalization_3 (Batch (None, 7, 7, 32)          128
_________________________________________________________________
activation_3 (Activation)    (None, 7, 7, 32)          0
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 3, 3, 32)          0
_________________________________________________________________
dropout_3 (Dropout)          (None, 3, 3, 32)          0
_________________________________________________________________
flatten_1 (Flatten)          (None, 288)               0
_________________________________________________________________
dense_1 (Dense)              (None, 256)               73984
_________________________________________________________________
dropout_4 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_2 (Dense)              (None, 5)                 1285
=================================================================
Total params: 105,285
Trainable params: 105,029
Non-trainable params: 256
_________________________________________________________________
```

3.3.2 For UCI dataset

On comparing the various supervised machine learning algorithms Random forest turned out to be the best classifier for the dataset.

3.4 Website

3.4.1 Frontend

The frontend of the web site was made using HTML/CSS. HTML is the standard mark-up language for Web pages. Cascading Style Sheets (CSS) is used to format the layout of a webpage. With CSS, you can control the colour, font, the size of text, the spacing between elements, how elements are positioned and laid out, what background images or background colours.

3.4.2 Backend

Flask: Flask is an API of Python that allows us to build up web-applications. It was developed by Armin Ronacher. Flask’s framework is more explicit than Django’s framework and is also easier to learn because it has less base code to implement a simple web-Application. A Web-Application Framework or Web Framework is the collection of modules and libraries that helps the developer to write applications without writing the low-level codes such as protocols, thread management, etc. Flask is based on WSGI (Web Server Gateway Interface) toolkit and Jinja2 template engine.

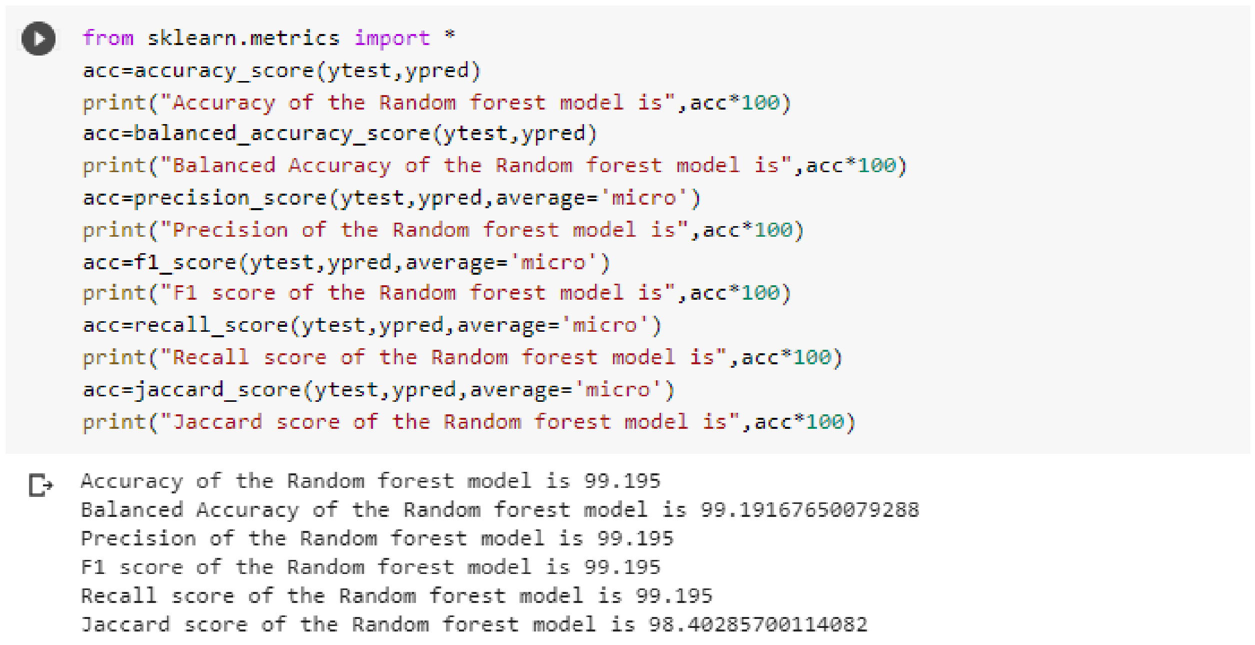
MySQL: MySQL is an open-source relational database management system (RDBMS).

4 Evaluation metrics

1. Accuracy = Number of correct predictions / Total number of predictions.
2. Balanced accuracy is calculated as the average of the proportion corrects of each class individually.
3. Recall gives the fraction you correctly identified as positive out of all positives.
4. Precision gives the fraction of correctly identified as positive out of all predicted as positives.
5. F1 score is defined as the harmonic mean of the model's precision and recall.
6. Jaccard similarity coefficient, defined as the size of the intersection divided by the size of the union of two label sets, is used to compare set of predicted labels for a sample to the corresponding set of labels in y_true.
7. A confusion matrix is a table that is often used to describe the performance of a classification model (or "classifier") on a set of test data for which the true values are known.
8. The classification report visualizer displays the precision, recall, F1, and support scores for the model.
9. An ROC curve (receiver operating characteristic curve) is a graph showing the performance of a classification model at all classification thresholds. This curve plots two parameters: True Positive Rate. False Positive Rate.
10. Mean squared error (MSE) or mean squared deviation (MSD) of an estimator (of a procedure for estimating an unobserved quantity) measures the average of the squares of the errors

4.1 Random forest

4.1.1 Accuracy, Balanced accuracy, F1 Score, Recall, Precision, Jaccard score

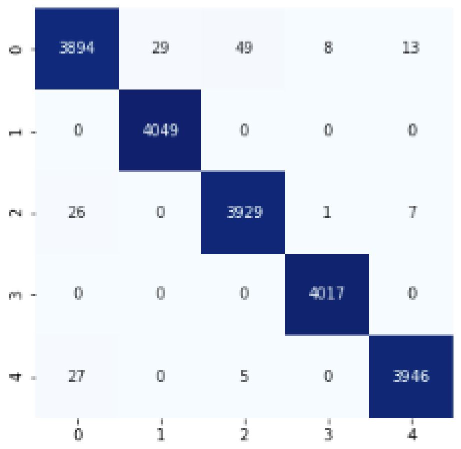


```
from sklearn.metrics import *
acc=accuracy_score(ytest,ypred)
print("Accuracy of the Random forest model is",acc*100)
acc=balanced_accuracy_score(ytest,ypred)
print("Balanced Accuracy of the Random forest model is",acc*100)
acc=precision_score(ytest,ypred,average='micro')
print("Precision of the Random forest model is",acc*100)
acc=f1_score(ytest,ypred,average='micro')
print("F1 score of the Random forest model is",acc*100)
acc=recall_score(ytest,ypred,average='micro')
print("Recall score of the Random forest model is",acc*100)
acc=jaccard_score(ytest,ypred,average='micro')
print("Jaccard score of the Random forest model is",acc*100)
```

```
Accuracy of the Random forest model is 99.195
Balanced Accuracy of the Random forest model is 99.19167650079288
Precision of the Random forest model is 99.195
F1 score of the Random forest model is 99.195
Recall score of the Random forest model is 99.195
Jaccard score of the Random forest model is 98.40285700114082
```

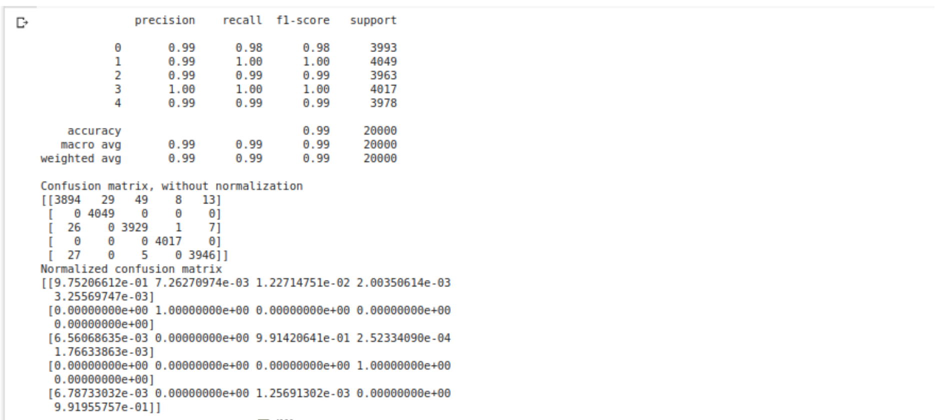
4.1.2 Confusion matrix

```
matplotlib.rcParams['figure.figsize']=(5,5)
from sklearn.metrics import confusion_matrix
cm=confusion_matrix(ytest,ypred)
sns.heatmap(cm,annot=True,fmt='d',cbar=False,cmap='Blues')
plt.show()
```

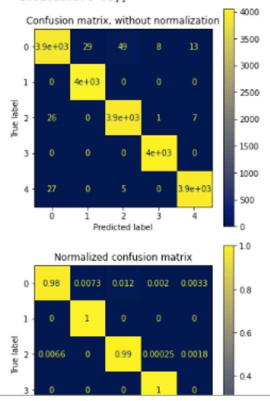


4.1.3 Normalized confusion matrix and Classification report

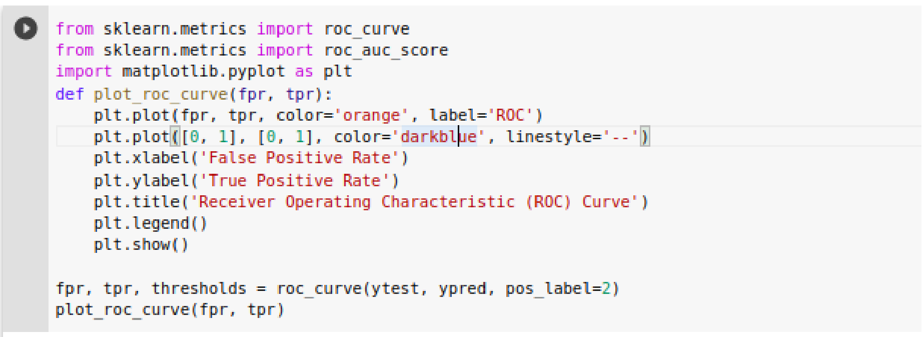
```
[45] print(classification_report(ytest,ypred))
     import matplotlib.pyplot as plt
     titles_options = [("Confusion matrix, without normalization", None),
                       ("Normalized confusion matrix", 'true')]
     for title, normalize in titles_options:
         disp = plot_confusion_matrix(ecg_mod, xtest, ytest,
                                      cmap=plt.cm.cividis,
                                      normalize=normalize)
         disp.ax_.set_title(title)

         print(title)
         print(disp.confusion_matrix)

     plt.show()
```



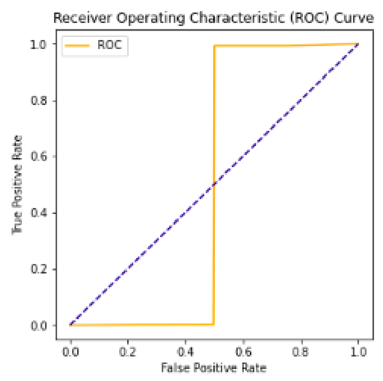
```
              precision    recall  f1-score   support

           0       0.99      0.98      0.98      3993
           1       0.99      1.00      1.00      4049
           2       0.99      0.99      0.99      3963
           3       1.00      1.00      1.00      4017
           4       0.99      0.99      0.99      3978

    accuracy                           0.99     20000
   macro avg       0.99      0.99      0.99     20000
weighted avg       0.99      0.99      0.99     20000

Confusion matrix, without normalization
[[3894   29   49    8   13]
 [   0 4049    0    0    0]
 [  26    0 3929    1    7]
 [   0    0    0 4017    0]
 [  27    0    5    0 3946]]
Normalized confusion matrix
[[9.75206612e-01 7.26270974e-03 1.22714751e-02 2.00350614e-03
  3.25569747e-03]
 [0.00000000e+00 1.00000000e+00 0.00000000e+00 0.00000000e+00
  0.00000000e+00]
 [6.56068635e-03 0.00000000e+00 9.91420641e-01 2.52334090e-04
  1.76633863e-03]
 [0.00000000e+00 0.00000000e+00 0.00000000e+00 1.00000000e+00
  0.00000000e+00]
 [6.78733032e-03 0.00000000e+00 1.25691302e-03 0.00000000e+00
  9.91955757e-01]]
```



4.1.4 ROC Curve

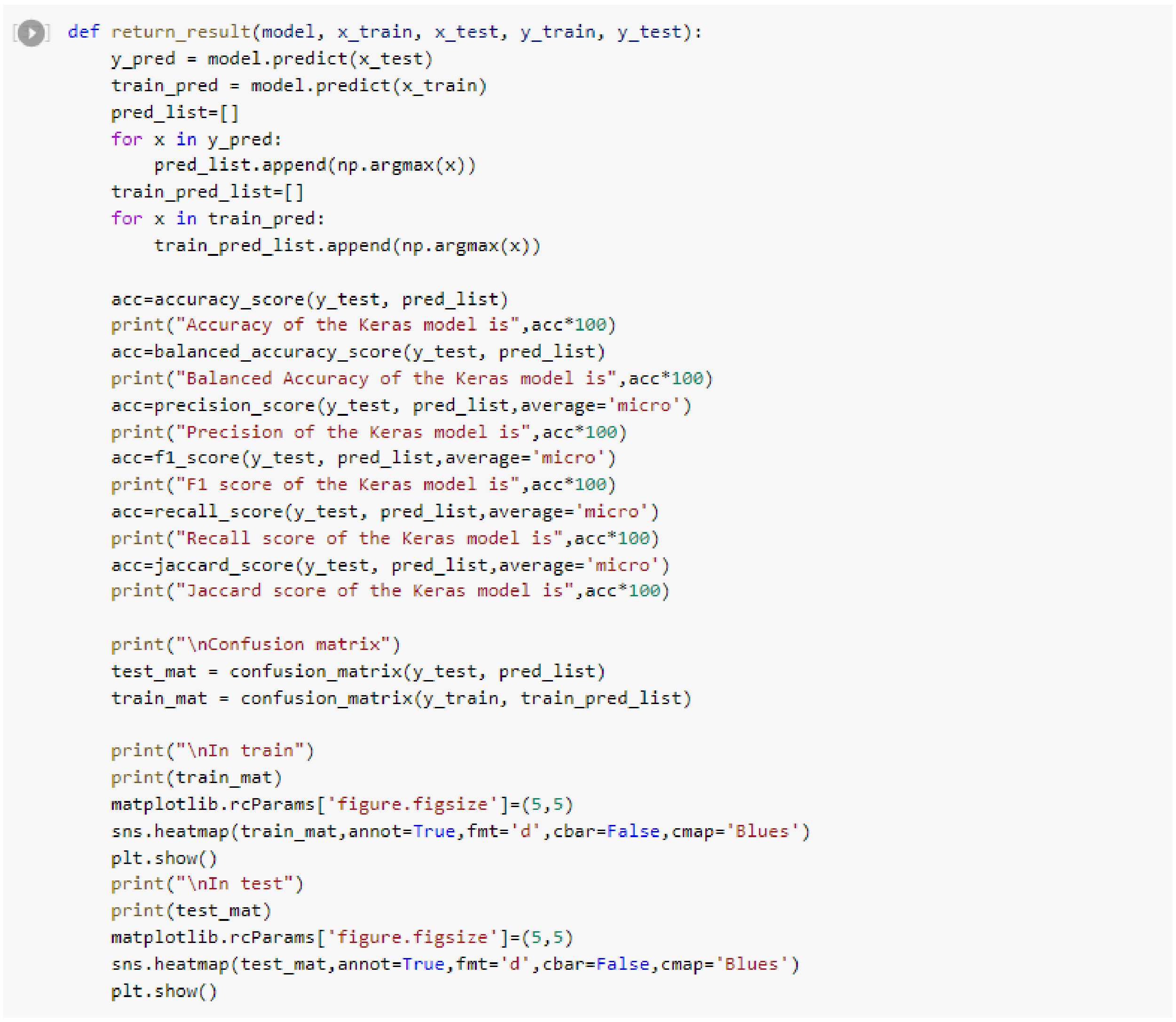


```python
from sklearn.metrics import roc_curve
from sklearn.metrics import roc_auc_score
import matplotlib.pyplot as plt
def plot_roc_curve(fpr, tpr):
    plt.plot(fpr, tpr, color='orange', label='ROC')
    plt.plot([0, 1], [0, 1], color='darkblue', linestyle='--')
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    plt.title('Receiver Operating Characteristic (ROC) Curve')
    plt.legend()
    plt.show()

fpr, tpr, thresholds = roc_curve(ytest, ypred, pos_label=2)
plot_roc_curve(fpr, tpr)
```

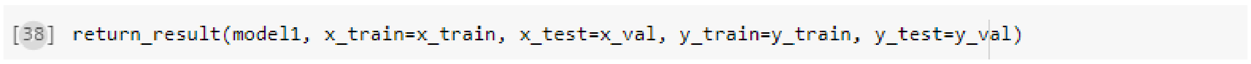


4.2 TensorFlow model

4.2.1 Accuracy, Balanced accuracy, F1 Score, Recall, Precision, Jaccard score, Confusion matrix

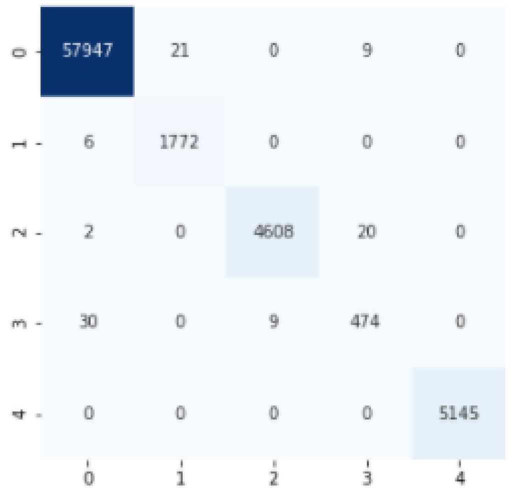


```
def return_result(model, x_train, x_test, y_train, y_test):
    y_pred = model.predict(x_test)
    train_pred = model.predict(x_train)
    pred_list=[]
    for x in y_pred:
        pred_list.append(np.argmax(x))
    train_pred_list=[]
    for x in train_pred:
        train_pred_list.append(np.argmax(x))

    acc=accuracy_score(y_test, pred_list)
    print("Accuracy of the Keras model is",acc*100)
    acc=balanced_accuracy_score(y_test, pred_list)
    print("Balanced Accuracy of the Keras model is",acc*100)
    acc=precision_score(y_test, pred_list,average='micro')
    print("Precision of the Keras model is",acc*100)
    acc=f1_score(y_test, pred_list,average='micro')
    print("F1 score of the Keras model is",acc*100)
    acc=recall_score(y_test, pred_list,average='micro')
    print("Recall score of the Keras model is",acc*100)
    acc=jaccard_score(y_test, pred_list,average='micro')
    print("Jaccard score of the Keras model is",acc*100)

    print("\nConfusion matrix")
    test_mat = confusion_matrix(y_test, pred_list)
    train_mat = confusion_matrix(y_train, train_pred_list)

    print("\nIn train")
    print(train_mat)
    matplotlib.rcParams['figure.figsize']=(5,5)
    sns.heatmap(train_mat,annot=True,fmt='d',cbar=False,cmap='Blues')
    plt.show()
    print("\nIn test")
    print(test_mat)
    matplotlib.rcParams['figure.figsize']=(5,5)
    sns.heatmap(test_mat,annot=True,fmt='d',cbar=False,cmap='Blues')
    plt.show()
```

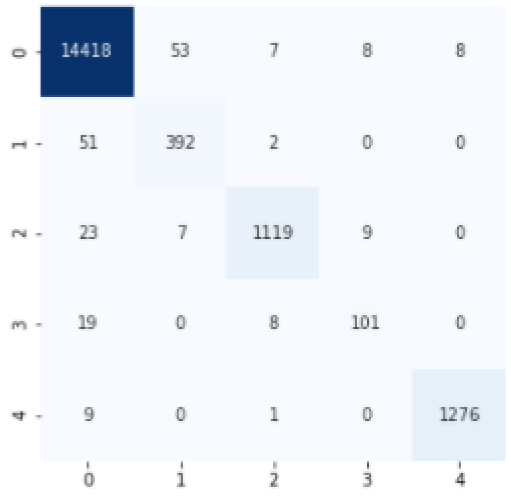


```
[38] return_result(model1, x_train=x_train, x_test=x_val, y_train=y_train, y_test=y_val)
```

```
Accuracy of the Keras model is 98.82930729255897
Balanced Accuracy of the Keras model is 92.46526042212622
Precision of the Keras model is 98.82930729255897
F1 score of the Keras model is 98.82930729255897
Recall score of the Keras model is 98.82930729255897
Jaccard score of the Keras model is 97.68570783472568

Confusion matrix

In train
[[57947    21     0     9     0]
 [    6  1772     0     0     0]
 [    2     0  4608    20     0]
 [   30     0     9   474     0]
 [    0     0     0     0  5145]]
```



```
In test
[[14418    53     7     8     8]
 [   51   392     2     0     0]
 [   23     7  1119     9     0]
 [   19     0     8   101     0]
 [    9     0     1     0  1276]]
```



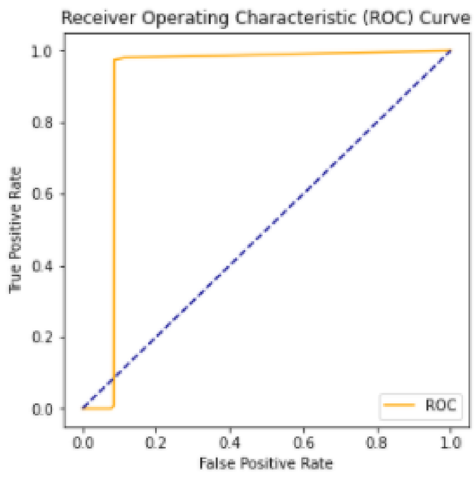
4.2.2 Classification report

```
[39] y_pred = model1.predict(x_val)
     train_pred = model1.predict(x_train)
     pred_list=[]
     for x in y_pred:
         pred_list.append(np.argmax(x))
     train_pred_list=[]
     for x in train_pred:
         train_pred_list.append(np.argmax(x))
     print(classification_report(y_val, pred_list))
```

```
              precision    recall  f1-score   support

         0.0       0.99      0.99      0.99     14494
         1.0       0.87      0.88      0.87       445
         2.0       0.98      0.97      0.98      1158
         3.0       0.86      0.79      0.82       128
         4.0       0.99      0.99      0.99      1286

    accuracy                           0.99     17511
   macro avg       0.94      0.92      0.93     17511
weighted avg       0.99      0.99      0.99     17511
```

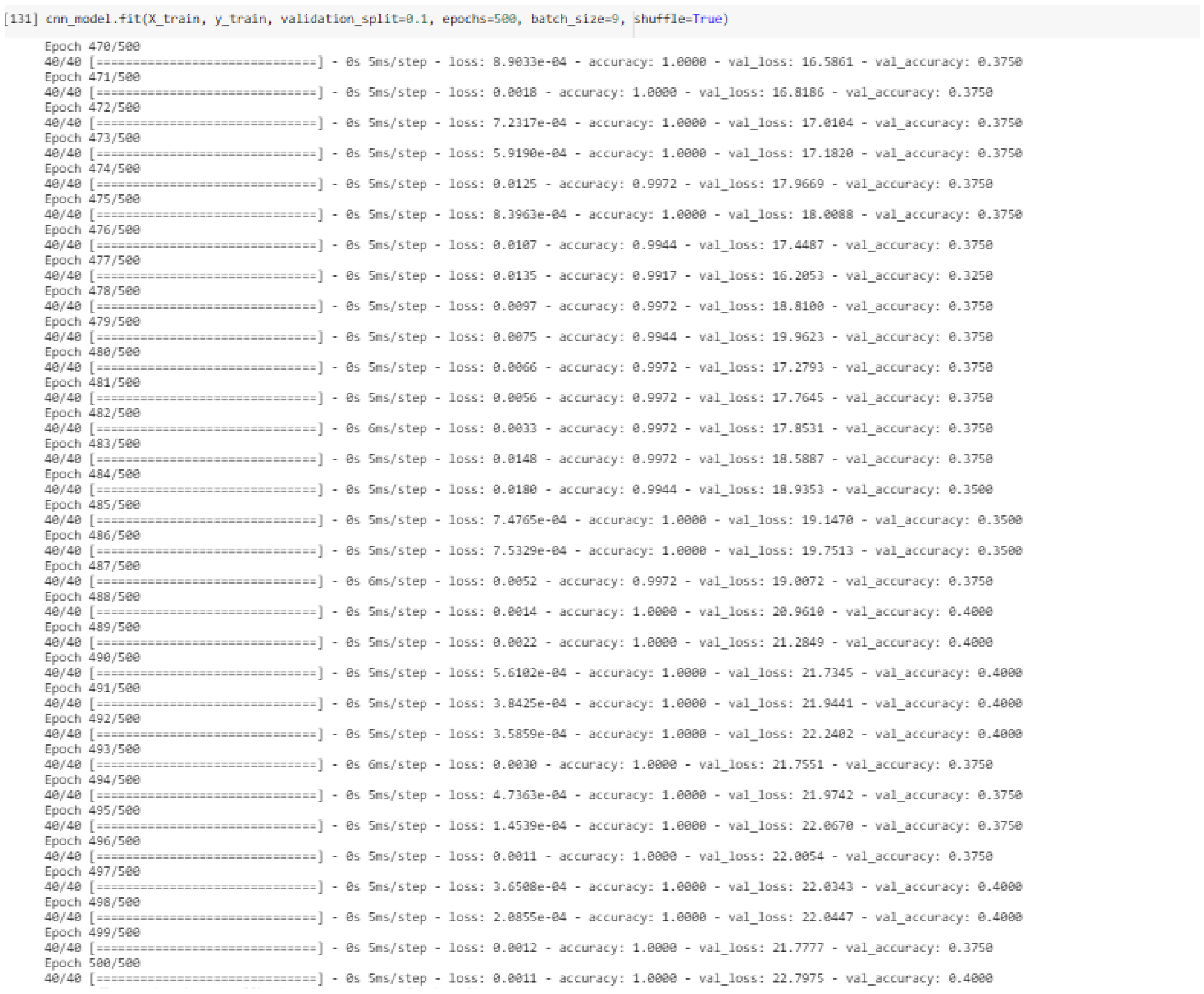
4.2.3 ROC Curve

```
[40] from sklearn.metrics import roc_curve
     from sklearn.metrics import roc_auc_score
     import matplotlib.pyplot as plt
     def plot_roc_curve(fpr, tpr):
         plt.plot(fpr, tpr, color='orange', label='ROC')
         plt.plot([0, 1], [0, 1], color='darkblue', linestyle='--')
         plt.xlabel('False Positive Rate')
         plt.ylabel('True Positive Rate')
         plt.title('Receiver Operating Characteristic (ROC) Curve')
         plt.legend()
         plt.show()

     fpr, tpr, thresholds = roc_curve(y_val, pred_list, pos_label=2)
     plot_roc_curve(fpr, tpr)
```



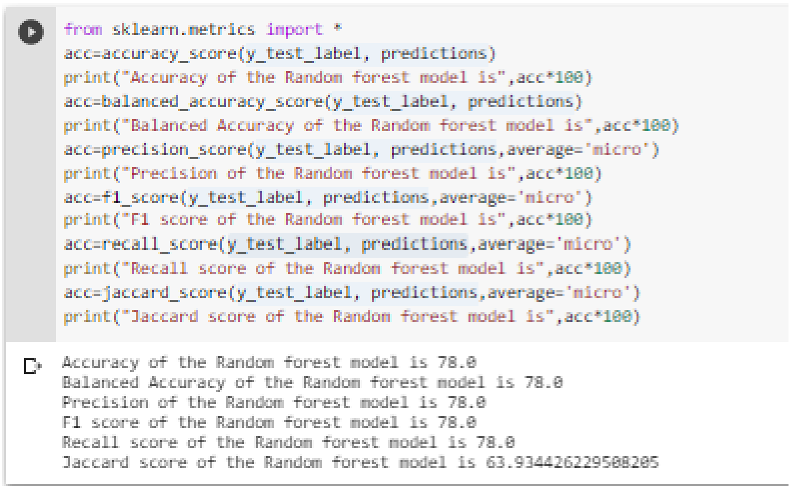
4.3 CNN model 1

4.3.1Accuracy, Value loss, Value Accuracy



```
[131] cnn_model.fit(X_train, y_train, validation_split=0.1, epochs=500, batch_size=9, shuffle=True)

      Epoch 470/500
      40/40 [==============================] - 0s 5ms/step - loss: 8.9033e-04 - accuracy: 1.0000 - val_loss: 16.5861 - val_accuracy: 0.3750
      Epoch 471/500
      40/40 [==============================] - 0s 5ms/step - loss: 0.0018 - accuracy: 1.0000 - val_loss: 16.8186 - val_accuracy: 0.3750
      Epoch 472/500
      40/40 [==============================] - 0s 5ms/step - loss: 7.2317e-04 - accuracy: 1.0000 - val_loss: 17.0104 - val_accuracy: 0.3750
      Epoch 473/500
      40/40 [==============================] - 0s 5ms/step - loss: 5.9190e-04 - accuracy: 1.0000 - val_loss: 17.1820 - val_accuracy: 0.3750
      Epoch 474/500
      40/40 [==============================] - 0s 5ms/step - loss: 0.0125 - accuracy: 0.9972 - val_loss: 17.9669 - val_accuracy: 0.3750
      Epoch 475/500
      40/40 [==============================] - 0s 5ms/step - loss: 8.3963e-04 - accuracy: 1.0000 - val_loss: 18.0088 - val_accuracy: 0.3750
      Epoch 476/500
      40/40 [==============================] - 0s 5ms/step - loss: 0.0107 - accuracy: 0.9944 - val_loss: 17.4487 - val_accuracy: 0.3750
      Epoch 477/500
      40/40 [==============================] - 0s 5ms/step - loss: 0.0135 - accuracy: 0.9917 - val_loss: 16.2053 - val_accuracy: 0.3250
      Epoch 478/500
      40/40 [==============================] - 0s 5ms/step - loss: 0.0097 - accuracy: 0.9972 - val_loss: 18.8100 - val_accuracy: 0.3750
      Epoch 479/500
      40/40 [==============================] - 0s 5ms/step - loss: 0.0075 - accuracy: 0.9944 - val_loss: 19.9623 - val_accuracy: 0.3750
      Epoch 480/500
      40/40 [==============================] - 0s 5ms/step - loss: 0.0066 - accuracy: 0.9972 - val_loss: 17.2793 - val_accuracy: 0.3750
      Epoch 481/500
      40/40 [==============================] - 0s 5ms/step - loss: 0.0056 - accuracy: 0.9972 - val_loss: 17.7645 - val_accuracy: 0.3750
      Epoch 482/500
      40/40 [==============================] - 0s 6ms/step - loss: 0.0033 - accuracy: 0.9972 - val_loss: 17.8531 - val_accuracy: 0.3750
      Epoch 483/500
      40/40 [==============================] - 0s 5ms/step - loss: 0.0148 - accuracy: 0.9972 - val_loss: 18.5887 - val_accuracy: 0.3750
      Epoch 484/500
      40/40 [==============================] - 0s 5ms/step - loss: 0.0180 - accuracy: 0.9944 - val_loss: 18.9353 - val_accuracy: 0.3500
      Epoch 485/500
      40/40 [==============================] - 0s 5ms/step - loss: 7.4765e-04 - accuracy: 1.0000 - val_loss: 19.1470 - val_accuracy: 0.3500
      Epoch 486/500
      40/40 [==============================] - 0s 5ms/step - loss: 7.5329e-04 - accuracy: 1.0000 - val_loss: 19.7513 - val_accuracy: 0.3500
      Epoch 487/500
      40/40 [==============================] - 0s 6ms/step - loss: 0.0052 - accuracy: 0.9972 - val_loss: 19.0072 - val_accuracy: 0.3750
      Epoch 488/500
      40/40 [==============================] - 0s 5ms/step - loss: 0.0014 - accuracy: 1.0000 - val_loss: 20.9610 - val_accuracy: 0.4000
      Epoch 489/500
      40/40 [==============================] - 0s 5ms/step - loss: 0.0022 - accuracy: 1.0000 - val_loss: 21.2849 - val_accuracy: 0.4000
      Epoch 490/500
      40/40 [==============================] - 0s 5ms/step - loss: 5.6102e-04 - accuracy: 1.0000 - val_loss: 21.7345 - val_accuracy: 0.4000
      Epoch 491/500
      40/40 [==============================] - 0s 5ms/step - loss: 3.8425e-04 - accuracy: 1.0000 - val_loss: 21.9441 - val_accuracy: 0.4000
      Epoch 492/500
      40/40 [==============================] - 0s 5ms/step - loss: 3.5859e-04 - accuracy: 1.0000 - val_loss: 22.2402 - val_accuracy: 0.4000
      Epoch 493/500
      40/40 [==============================] - 0s 6ms/step - loss: 0.0030 - accuracy: 1.0000 - val_loss: 21.7551 - val_accuracy: 0.3750
      Epoch 494/500
      40/40 [==============================] - 0s 5ms/step - loss: 4.7363e-04 - accuracy: 1.0000 - val_loss: 21.9742 - val_accuracy: 0.3750
      Epoch 495/500
      40/40 [==============================] - 0s 5ms/step - loss: 1.4539e-04 - accuracy: 1.0000 - val_loss: 22.0670 - val_accuracy: 0.3750
      Epoch 496/500
      40/40 [==============================] - 0s 5ms/step - loss: 0.0011 - accuracy: 1.0000 - val_loss: 22.0054 - val_accuracy: 0.3750
      Epoch 497/500
      40/40 [==============================] - 0s 5ms/step - loss: 3.6508e-04 - accuracy: 1.0000 - val_loss: 22.0343 - val_accuracy: 0.4000
      Epoch 498/500
      40/40 [==============================] - 0s 5ms/step - loss: 2.0855e-04 - accuracy: 1.0000 - val_loss: 22.0447 - val_accuracy: 0.4000
      Epoch 499/500
      40/40 [==============================] - 0s 5ms/step - loss: 0.0012 - accuracy: 1.0000 - val_loss: 21.7777 - val_accuracy: 0.3750
      Epoch 500/500
      40/40 [==============================] - 0s 5ms/step - loss: 0.0011 - accuracy: 1.0000 - val_loss: 22.7975 - val_accuracy: 0.4000
```

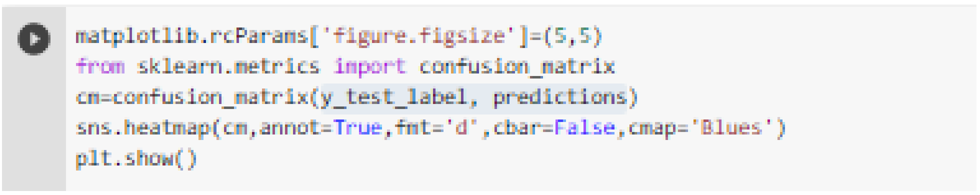
4.3.2 Accuracy, Balanced accuracy, F1 Score, Recall, Precision, Jaccard score



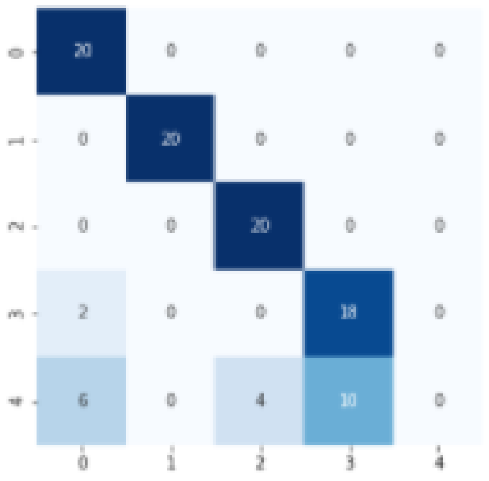
```
from sklearn.metrics import *
acc=accuracy_score(y_test_label, predictions)
print("Accuracy of the Random forest model is",acc*100)
acc=balanced_accuracy_score(y_test_label, predictions)
print("Balanced Accuracy of the Random forest model is",acc*100)
acc=precision_score(y_test_label, predictions,average='micro')
print("Precision of the Random forest model is",acc*100)
acc=f1_score(y_test_label, predictions,average='micro')
print("F1 score of the Random forest model is",acc*100)
acc=recall_score(y_test_label, predictions,average='micro')
print("Recall score of the Random forest model is",acc*100)
acc=jaccard_score(y_test_label, predictions,average='micro')
print("Jaccard score of the Random forest model is",acc*100)
```

```
Accuracy of the Random forest model is 78.0
Balanced Accuracy of the Random forest model is 78.0
Precision of the Random forest model is 78.0
F1 score of the Random forest model is 78.0
Recall score of the Random forest model is 78.0
Jaccard score of the Random forest model is 63.934426229508205
```

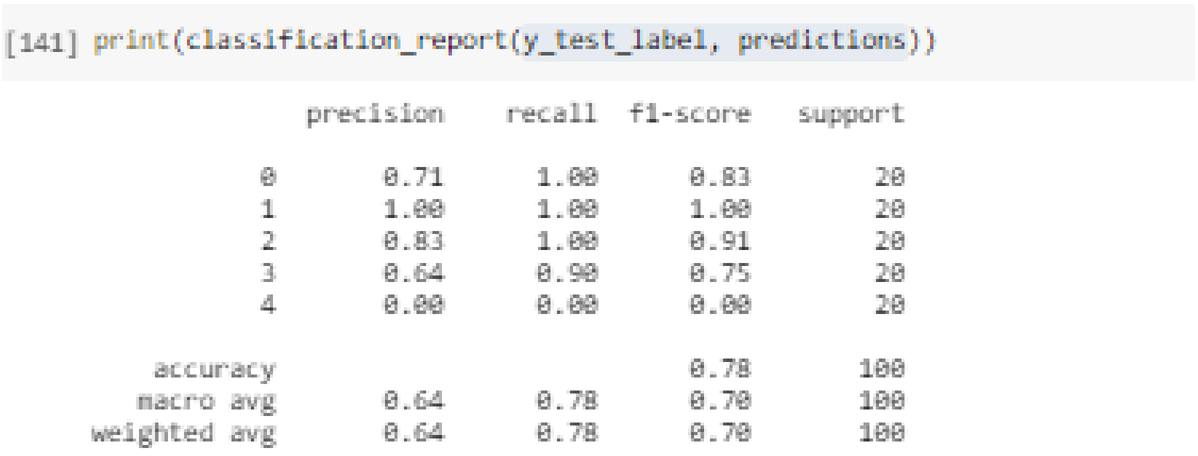
4.3.3 Confusion Matrix



```
matplotlib.rcParams['figure.figsize']=(5,5)
from sklearn.metrics import confusion_matrix
cm=confusion_matrix(y_test_label, predictions)
sns.heatmap(cm,annot=True,fmt='d',cbar=False,cmap='Blues')
plt.show()
```



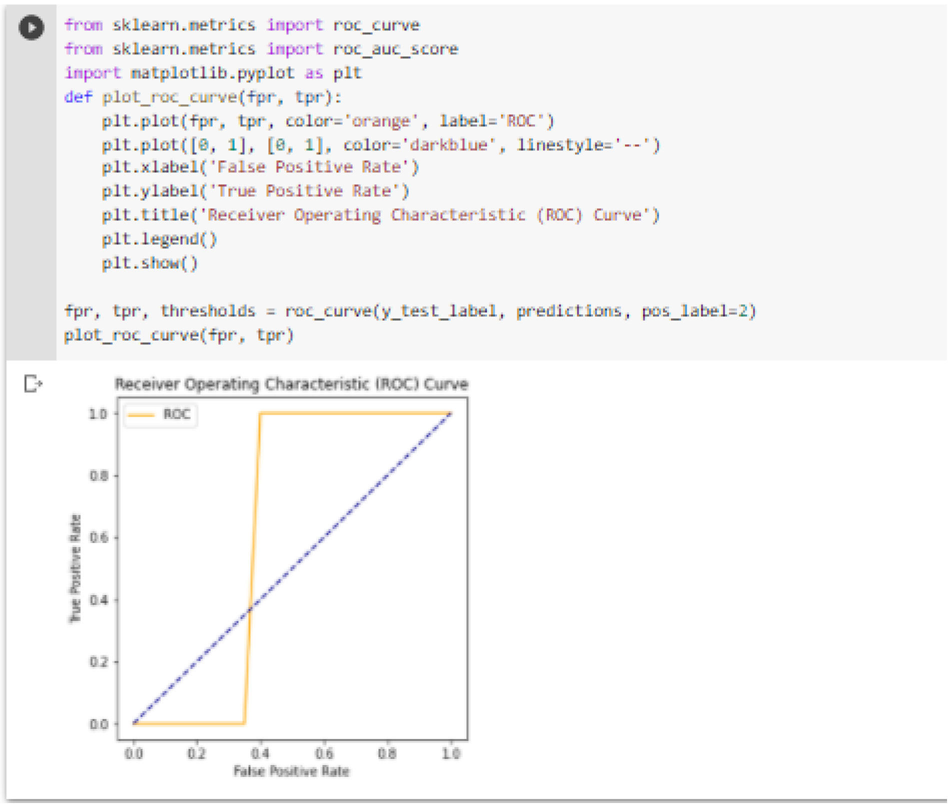
4.3.4 Classification Report



```
[141] print(classification_report(y_test_label, predictions))
```

```
              precision    recall  f1-score   support

           0       0.71      1.00      0.83        20
           1       1.00      1.00      1.00        20
           2       0.83      1.00      0.91        20
           3       0.64      0.90      0.75        20
           4       0.00      0.00      0.00        20

    accuracy                           0.78       100
   macro avg       0.64      0.78      0.70       100
weighted avg       0.64      0.78      0.70       100
```

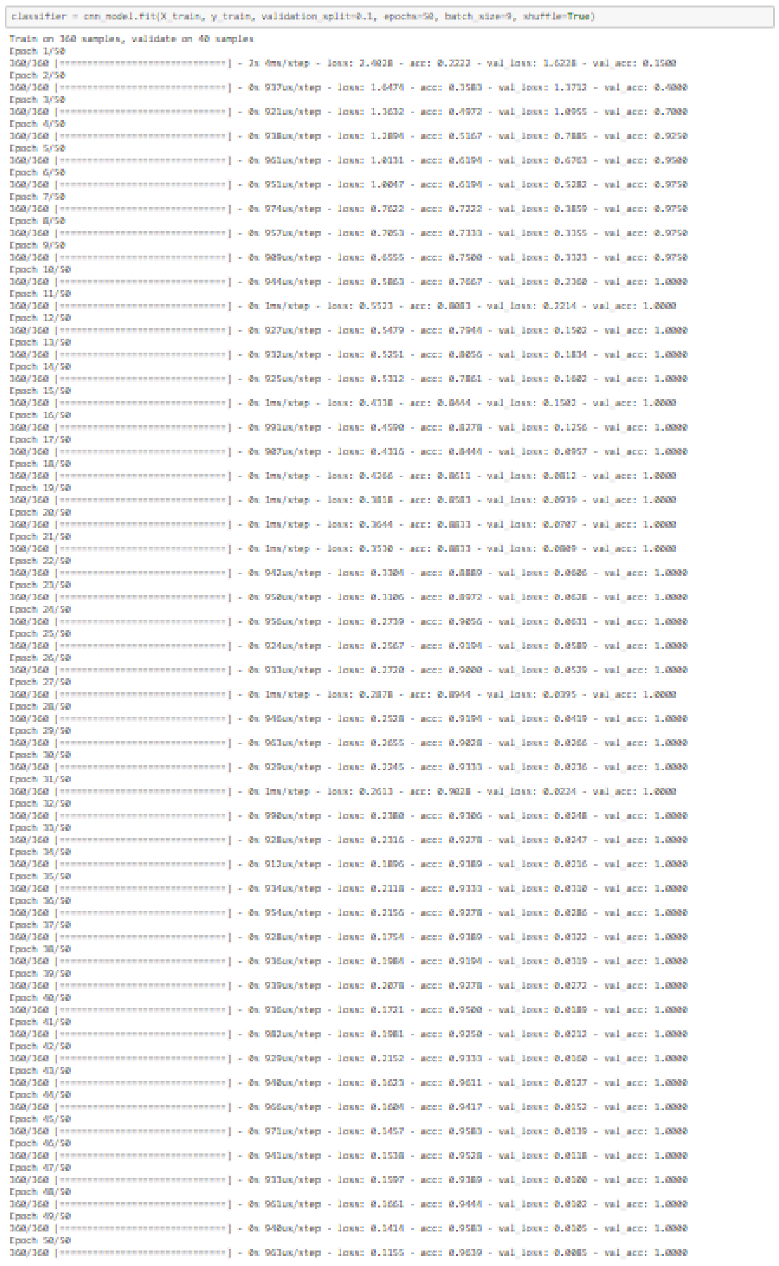
4.3.5 ROC curve



```
from sklearn.metrics import roc_curve
from sklearn.metrics import roc_auc_score
import matplotlib.pyplot as plt
def plot_roc_curve(fpr, tpr):
    plt.plot(fpr, tpr, color='orange', label='ROC')
    plt.plot([0, 1], [0, 1], color='darkblue', linestyle='--')
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    plt.title('Receiver Operating Characteristic (ROC) Curve')
    plt.legend()
    plt.show()

fpr, tpr, thresholds = roc_curve(y_test_label, predictions, pos_label=2)
plot_roc_curve(fpr, tpr)
```

4.4 CNN model 2

4.4.1 Accuracy, Value loss, Value Accuracy



```
classifier = cnn_model.fit(X_train, y_train, validation_split=0.1, epochs=50, batch_size=9, shuffle=True)
```

```
Train on 360 samples, validate on 40 samples
Epoch 1/50
360/360 [==============================] - 2s 4ms/step - loss: 2.4028 - acc: 0.2222 - val_loss: 1.6228 - val_acc: 0.1500
Epoch 2/50
360/360 [==============================] - 0s 937us/step - loss: 1.6474 - acc: 0.3583 - val_loss: 1.3712 - val_acc: 0.4000
Epoch 3/50
360/360 [==============================] - 0s 921us/step - loss: 1.3632 - acc: 0.4972 - val_loss: 1.0955 - val_acc: 0.7000
Epoch 4/50
360/360 [==============================] - 0s 938us/step - loss: 1.2894 - acc: 0.5167 - val_loss: 0.7885 - val_acc: 0.9250
Epoch 5/50
360/360 [==============================] - 0s 961us/step - loss: 1.0131 - acc: 0.6194 - val_loss: 0.6763 - val_acc: 0.9500
Epoch 6/50
360/360 [==============================] - 0s 951us/step - loss: 1.0047 - acc: 0.6194 - val_loss: 0.5282 - val_acc: 0.9750
Epoch 7/50
360/360 [==============================] - 0s 974us/step - loss: 0.7622 - acc: 0.7222 - val_loss: 0.3859 - val_acc: 0.9750
Epoch 8/50
360/360 [==============================] - 0s 957us/step - loss: 0.7053 - acc: 0.7333 - val_loss: 0.3355 - val_acc: 0.9750
Epoch 9/50
360/360 [==============================] - 0s 989us/step - loss: 0.6555 - acc: 0.7500 - val_loss: 0.3323 - val_acc: 0.9750
Epoch 10/50
360/360 [==============================] - 0s 944us/step - loss: 0.5863 - acc: 0.7667 - val_loss: 0.2360 - val_acc: 1.0000
Epoch 11/50
360/360 [==============================] - 0s 1ms/step - loss: 0.5533 - acc: 0.8083 - val_loss: 0.2214 - val_acc: 1.0000
Epoch 12/50
360/360 [==============================] - 0s 927us/step - loss: 0.5479 - acc: 0.7944 - val_loss: 0.1582 - val_acc: 1.0000
Epoch 13/50
360/360 [==============================] - 0s 932us/step - loss: 0.5251 - acc: 0.8056 - val_loss: 0.1834 - val_acc: 1.0000
Epoch 14/50
360/360 [==============================] - 0s 925us/step - loss: 0.5312 - acc: 0.7861 - val_loss: 0.1602 - val_acc: 1.0000
Epoch 15/50
360/360 [==============================] - 0s 1ms/step - loss: 0.4318 - acc: 0.8444 - val_loss: 0.1583 - val_acc: 1.0000
Epoch 16/50
360/360 [==============================] - 0s 991us/step - loss: 0.4590 - acc: 0.8278 - val_loss: 0.1256 - val_acc: 1.0000
Epoch 17/50
360/360 [==============================] - 0s 987us/step - loss: 0.4316 - acc: 0.8444 - val_loss: 0.0957 - val_acc: 1.0000
Epoch 18/50
360/360 [==============================] - 0s 1ms/step - loss: 0.4266 - acc: 0.8611 - val_loss: 0.0812 - val_acc: 1.0000
Epoch 19/50
360/360 [==============================] - 0s 1ms/step - loss: 0.3818 - acc: 0.8583 - val_loss: 0.0939 - val_acc: 1.0000
Epoch 20/50
360/360 [==============================] - 0s 1ms/step - loss: 0.3644 - acc: 0.8833 - val_loss: 0.0707 - val_acc: 1.0000
Epoch 21/50
360/360 [==============================] - 0s 1ms/step - loss: 0.3530 - acc: 0.8833 - val_loss: 0.0809 - val_acc: 1.0000
Epoch 22/50
360/360 [==============================] - 0s 942us/step - loss: 0.3304 - acc: 0.8889 - val_loss: 0.0606 - val_acc: 1.0000
Epoch 23/50
360/360 [==============================] - 0s 950us/step - loss: 0.3106 - acc: 0.8972 - val_loss: 0.0628 - val_acc: 1.0000
Epoch 24/50
360/360 [==============================] - 0s 956us/step - loss: 0.2739 - acc: 0.9056 - val_loss: 0.0631 - val_acc: 1.0000
Epoch 25/50
360/360 [==============================] - 0s 924us/step - loss: 0.2567 - acc: 0.9194 - val_loss: 0.0589 - val_acc: 1.0000
Epoch 26/50
360/360 [==============================] - 0s 933us/step - loss: 0.2720 - acc: 0.9000 - val_loss: 0.0539 - val_acc: 1.0000
Epoch 27/50
360/360 [==============================] - 0s 1ms/step - loss: 0.2878 - acc: 0.8944 - val_loss: 0.0395 - val_acc: 1.0000
Epoch 28/50
360/360 [==============================] - 0s 946us/step - loss: 0.2528 - acc: 0.9194 - val_loss: 0.0419 - val_acc: 1.0000
Epoch 29/50
360/360 [==============================] - 0s 963us/step - loss: 0.2655 - acc: 0.9028 - val_loss: 0.0266 - val_acc: 1.0000
Epoch 30/50
360/360 [==============================] - 0s 929us/step - loss: 0.2245 - acc: 0.9333 - val_loss: 0.0236 - val_acc: 1.0000
Epoch 31/50
360/360 [==============================] - 0s 1ms/step - loss: 0.2613 - acc: 0.9028 - val_loss: 0.0224 - val_acc: 1.0000
Epoch 32/50
360/360 [==============================] - 0s 990us/step - loss: 0.2380 - acc: 0.9306 - val_loss: 0.0248 - val_acc: 1.0000
Epoch 33/50
360/360 [==============================] - 0s 928us/step - loss: 0.2316 - acc: 0.9278 - val_loss: 0.0247 - val_acc: 1.0000
Epoch 34/50
360/360 [==============================] - 0s 912us/step - loss: 0.1896 - acc: 0.9389 - val_loss: 0.0216 - val_acc: 1.0000
Epoch 35/50
360/360 [==============================] - 0s 934us/step - loss: 0.2118 - acc: 0.9333 - val_loss: 0.0310 - val_acc: 1.0000
Epoch 36/50
360/360 [==============================] - 0s 954us/step - loss: 0.2156 - acc: 0.9278 - val_loss: 0.0286 - val_acc: 1.0000
Epoch 37/50
360/360 [==============================] - 0s 928us/step - loss: 0.1754 - acc: 0.9389 - val_loss: 0.0322 - val_acc: 1.0000
Epoch 38/50
360/360 [==============================] - 0s 936us/step - loss: 0.1984 - acc: 0.9194 - val_loss: 0.0319 - val_acc: 1.0000
Epoch 39/50
360/360 [==============================] - 0s 939us/step - loss: 0.2078 - acc: 0.9278 - val_loss: 0.0272 - val_acc: 1.0000
Epoch 40/50
360/360 [==============================] - 0s 936us/step - loss: 0.1721 - acc: 0.9500 - val_loss: 0.0189 - val_acc: 1.0000
Epoch 41/50
360/360 [==============================] - 0s 982us/step - loss: 0.1981 - acc: 0.9250 - val_loss: 0.0212 - val_acc: 1.0000
Epoch 42/50
360/360 [==============================] - 0s 929us/step - loss: 0.2152 - acc: 0.9333 - val_loss: 0.0160 - val_acc: 1.0000
Epoch 43/50
360/360 [==============================] - 0s 940us/step - loss: 0.1623 - acc: 0.9611 - val_loss: 0.0127 - val_acc: 1.0000
Epoch 44/50
360/360 [==============================] - 0s 966us/step - loss: 0.1604 - acc: 0.9417 - val_loss: 0.0152 - val_acc: 1.0000
Epoch 45/50
360/360 [==============================] - 0s 971us/step - loss: 0.1457 - acc: 0.9583 - val_loss: 0.0139 - val_acc: 1.0000
Epoch 46/50
360/360 [==============================] - 0s 941us/step - loss: 0.1538 - acc: 0.9528 - val_loss: 0.0118 - val_acc: 1.0000
Epoch 47/50
360/360 [==============================] - 0s 933us/step - loss: 0.1597 - acc: 0.9389 - val_loss: 0.0100 - val_acc: 1.0000
Epoch 48/50
360/360 [==============================] - 0s 961us/step - loss: 0.1661 - acc: 0.9444 - val_loss: 0.0102 - val_acc: 1.0000
Epoch 49/50
360/360 [==============================] - 0s 940us/step - loss: 0.1414 - acc: 0.9583 - val_loss: 0.0105 - val_acc: 1.0000
Epoch 50/50
360/360 [==============================] - 0s 963us/step - loss: 0.1155 - acc: 0.9639 - val_loss: 0.0085 - val_acc: 1.0000
```

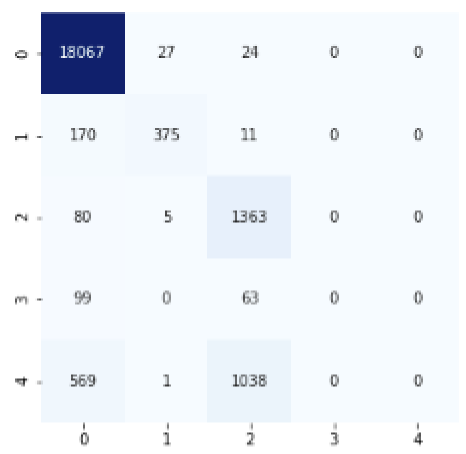
4.4.2 Accuracy, Balanced Accuracy, F1 Score, Recall

```
[80] from sklearn.metrics import *
     acc=accuracy_score(Y_test, pred_test)
     print("Accuracy of the CNN model is",acc*100)
     acc=balanced_accuracy_score(Y_test, pred_test)
     print("Balanced Accuracy of the CNN model is",acc*100)
     acc=f1_score(Y_test, pred_test,average='micro')
     print("F1 score of the CNN model is",acc*100)
     acc=recall_score(Y_test, pred_test,average='micro')
     print("Recall score of the CNN model is",acc*100)
```

```
Accuracy of the CNN model is 90.46683720080395
Balanced Accuracy of the CNN model is 52.258877879330136
F1 score of the CNN model is 90.46683720080395
Recall score of the CNN model is 90.46683720080395
```

4.4.3 Confusion matrix

```
matplotlib.rcParams['figure.figsize']=(5,5)
from sklearn.metrics import confusion_matrix
cm=confusion_matrix(Y_test, pred_test)
sns.heatmap(cm,annot=True,fmt='d',cbar=False,cmap='Blues')
plt.show()
```



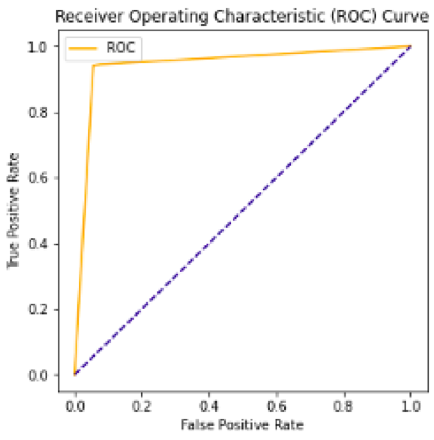
4.4.4 Classification report

```
print(classification_report(Y_test, pred_test))
```

```
              precision    recall  f1-score   support

           0       0.95      1.00      0.97     18118
           1       0.92      0.67      0.78       556
           2       0.55      0.94      0.69      1448
           3       0.00      0.00      0.00       162
           4       0.00      0.00      0.00      1608

    accuracy                           0.90     21892
   macro avg       0.48      0.52      0.49     21892
weighted avg       0.85      0.90      0.87     21892
```

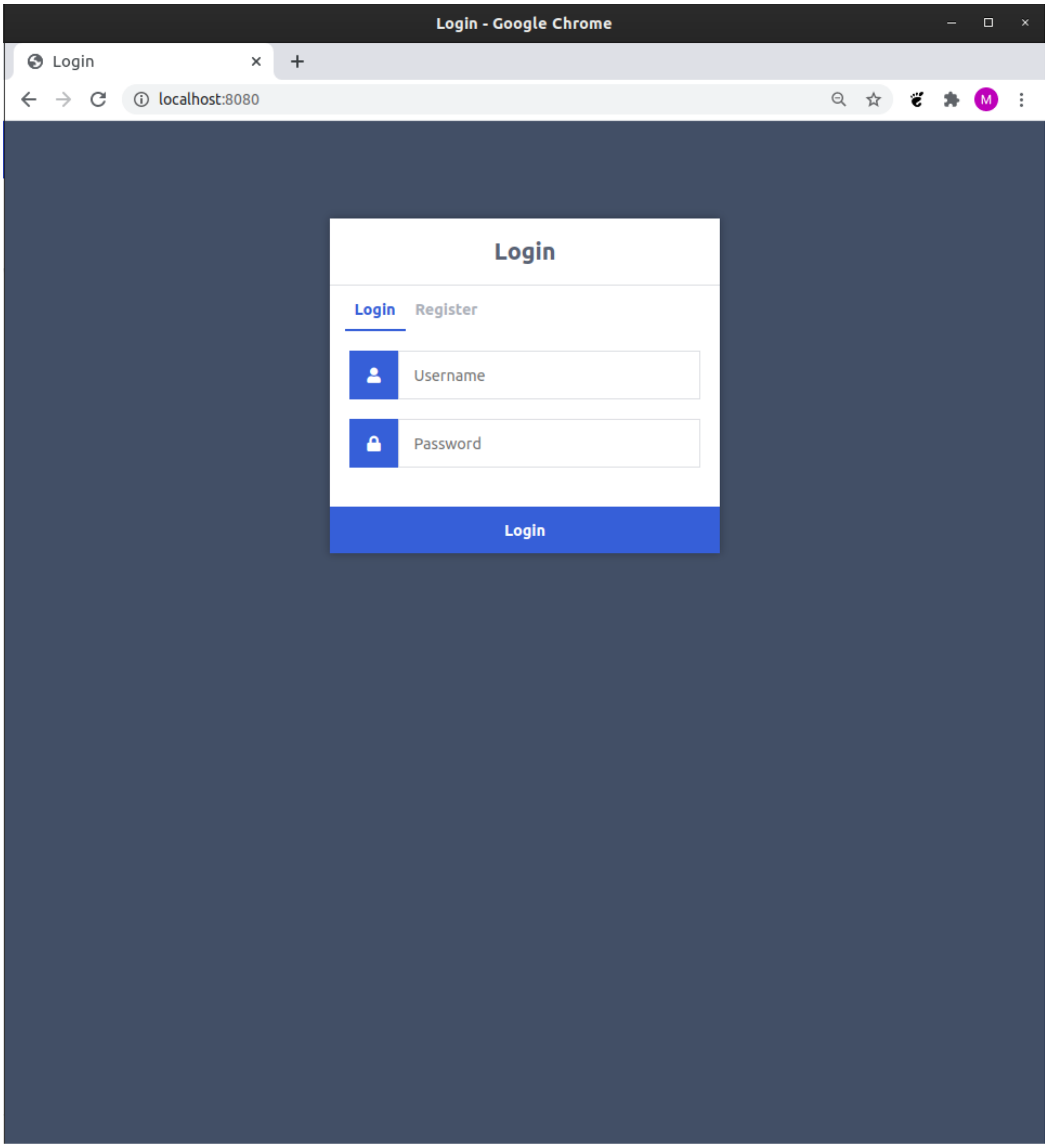
4.4.5 ROC curve

```
[88] from sklearn.metrics import roc_curve
     from sklearn.metrics import roc_auc_score
     import matplotlib.pyplot as plt
     def plot_roc_curve(fpr, tpr):
         plt.plot(fpr, tpr, color='orange', label='ROC')
         plt.plot([0, 1], [0, 1], color='darkblue', linestyle='--')
         plt.xlabel('False Positive Rate')
         plt.ylabel('True Positive Rate')
         plt.title('Receiver Operating Characteristic (ROC) Curve')
         plt.legend()
         plt.show()

     fpr, tpr, thresholds = roc_curve(Y_test, pred_test, pos_label=2)
     plot_roc_curve(fpr, tpr)
```

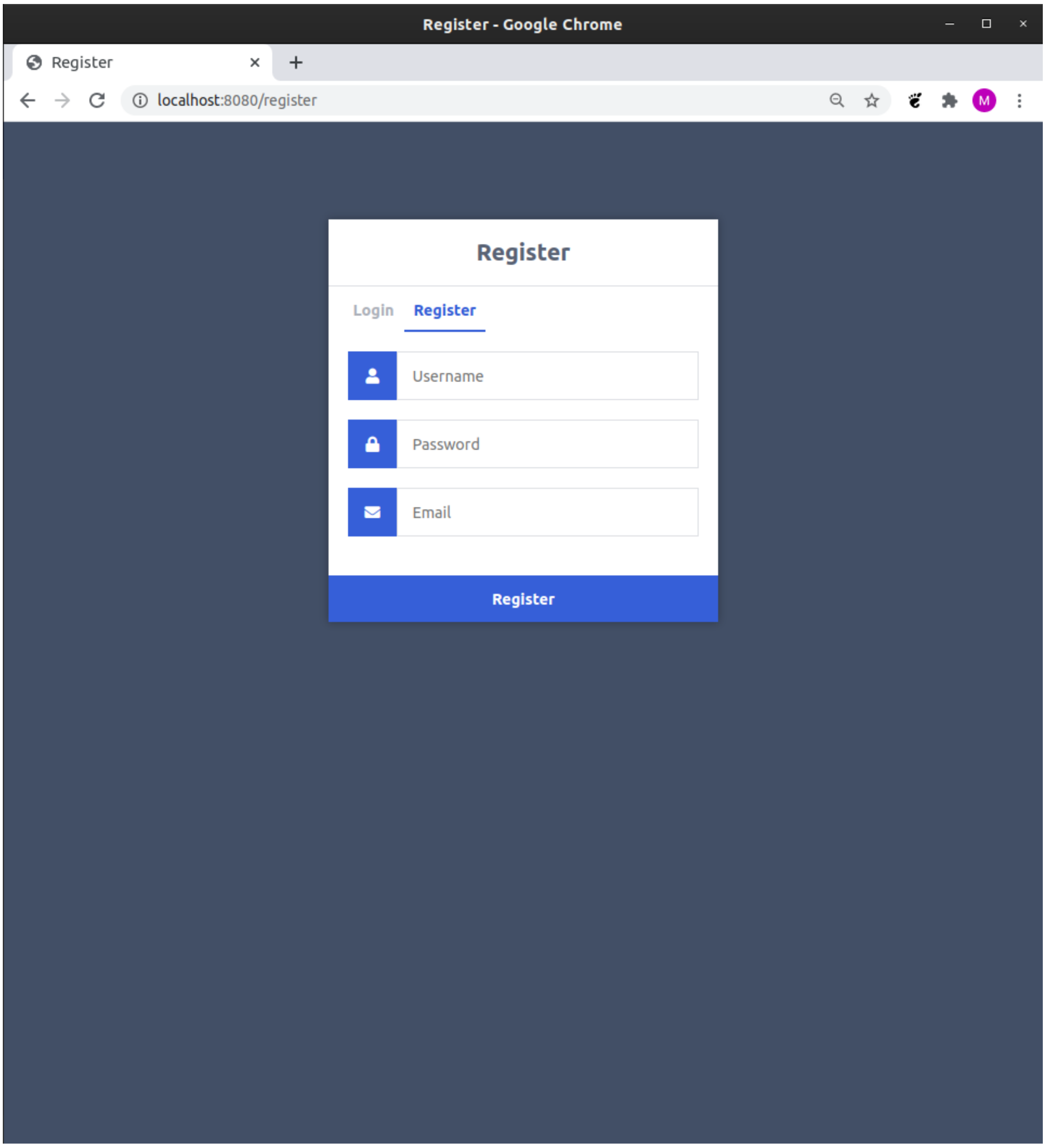


5 Screenshots of Application output

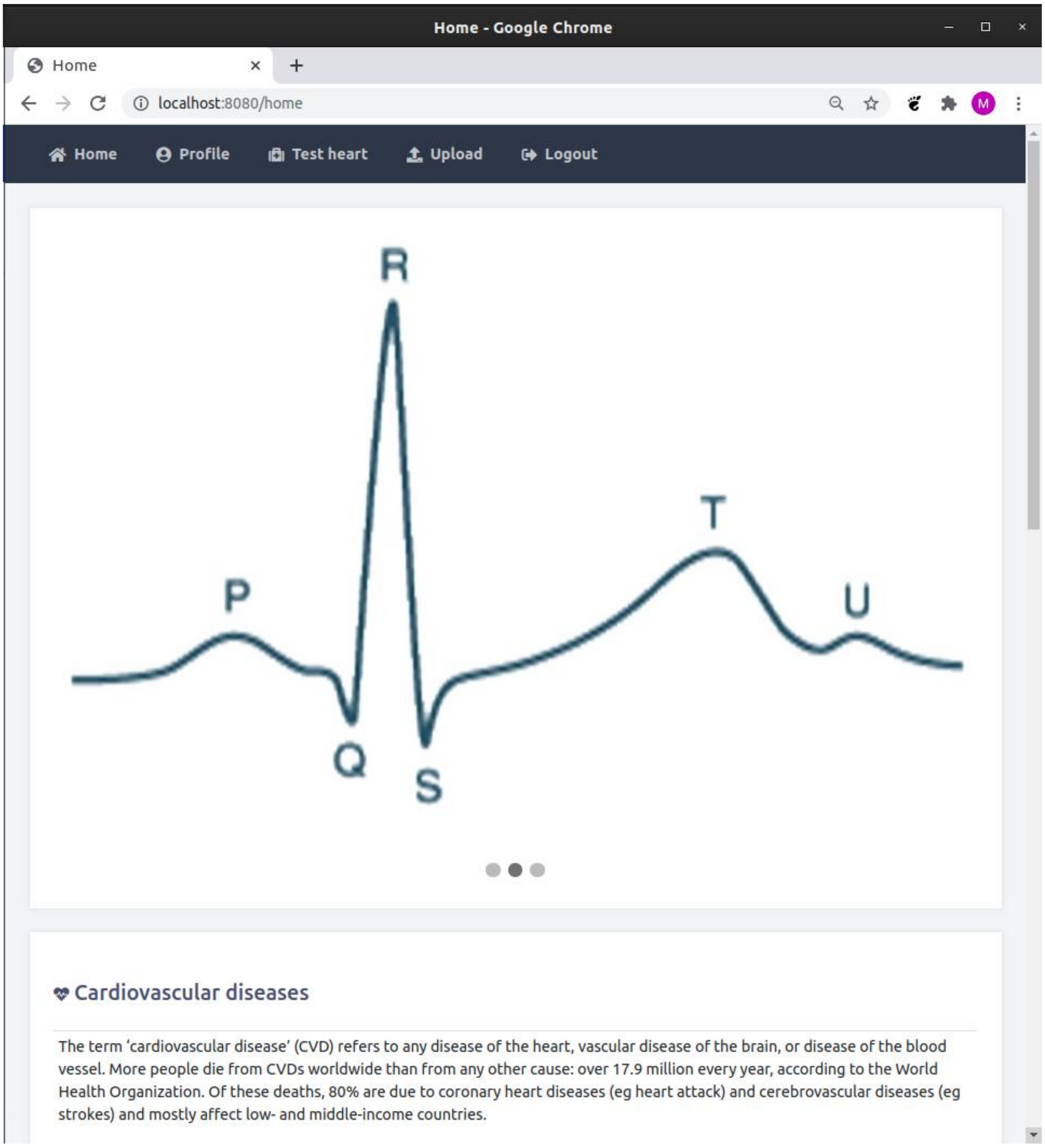
5.1 Login



5.2 Register


Register - Google Chrome
Register
localhost:8080/register
Register
Login
Register
Username
Password
Email
Register

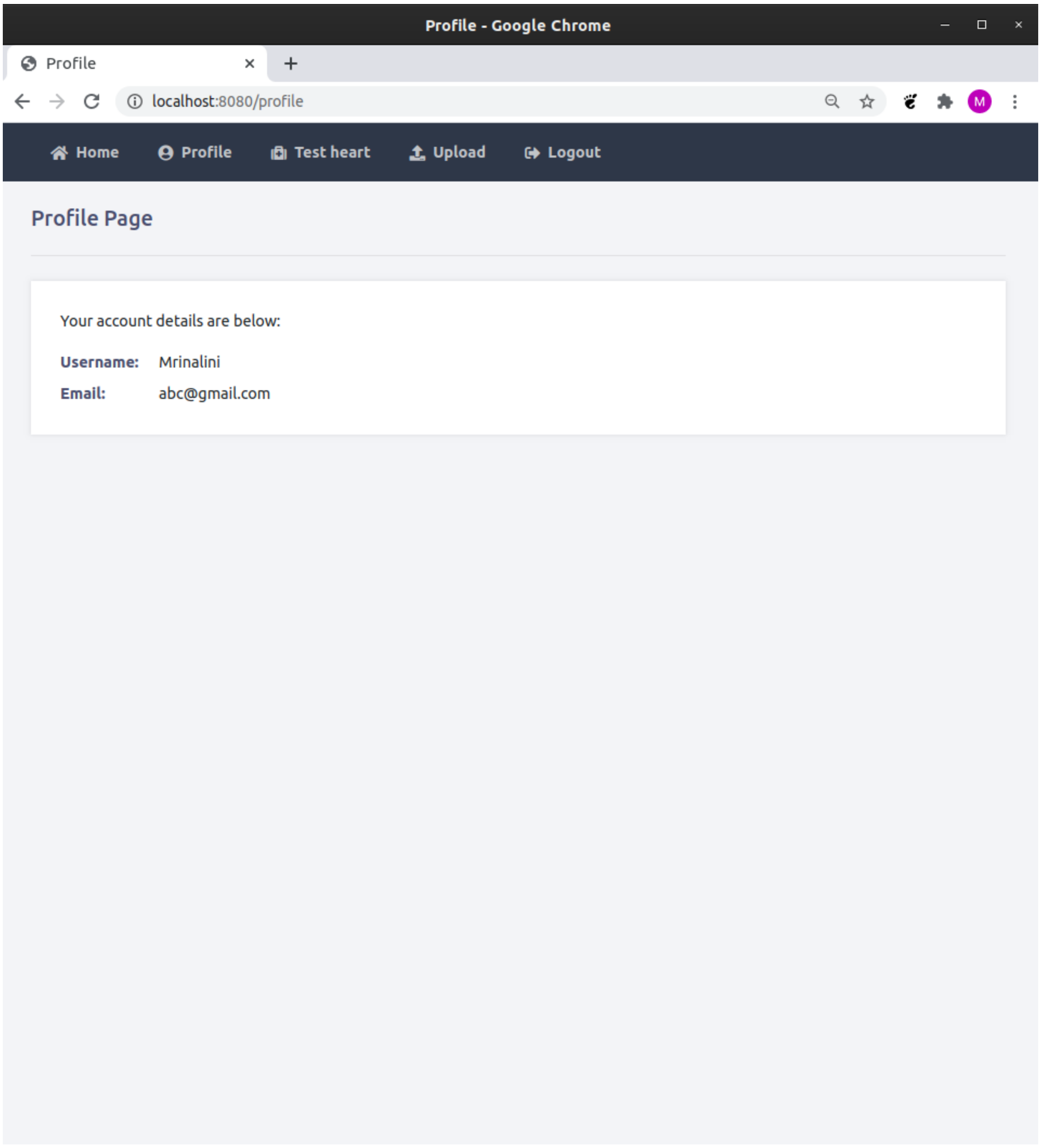
5.3 Home page



Cardiovascular diseases

The term ‘cardiovascular disease’ (CVD) refers to any disease of the heart, vascular disease of the brain, or disease of the blood vessel. More people die from CVDs worldwide than from any other cause: over 17.9 million every year, according to the World Health Organization. Of these deaths, 80% are due to coronary heart diseases (eg heart attack) and cerebrovascular diseases (eg strokes) and mostly affect low- and middle-income countries.

5.4 Profile



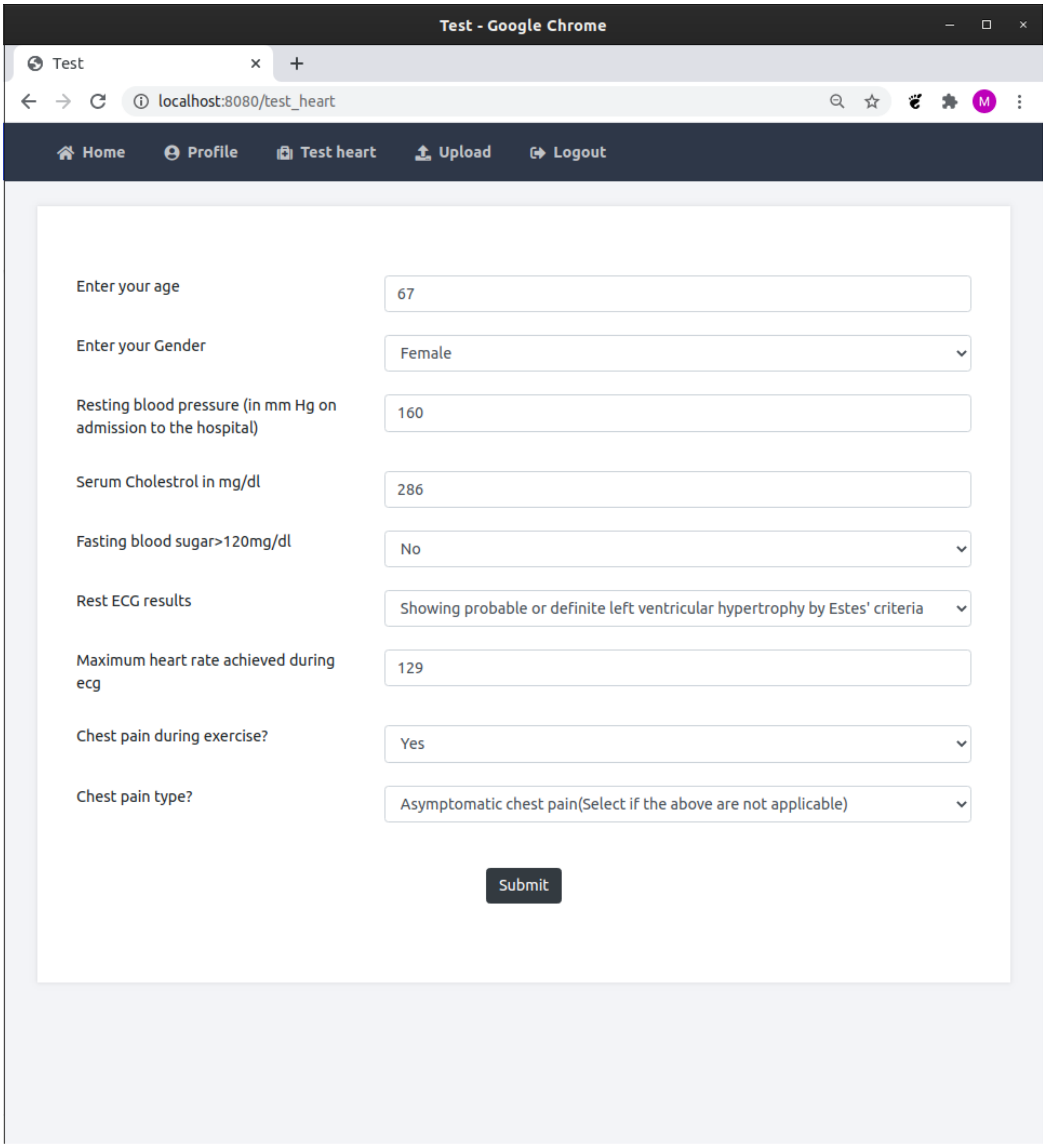
Profile Page

Your account details are below:

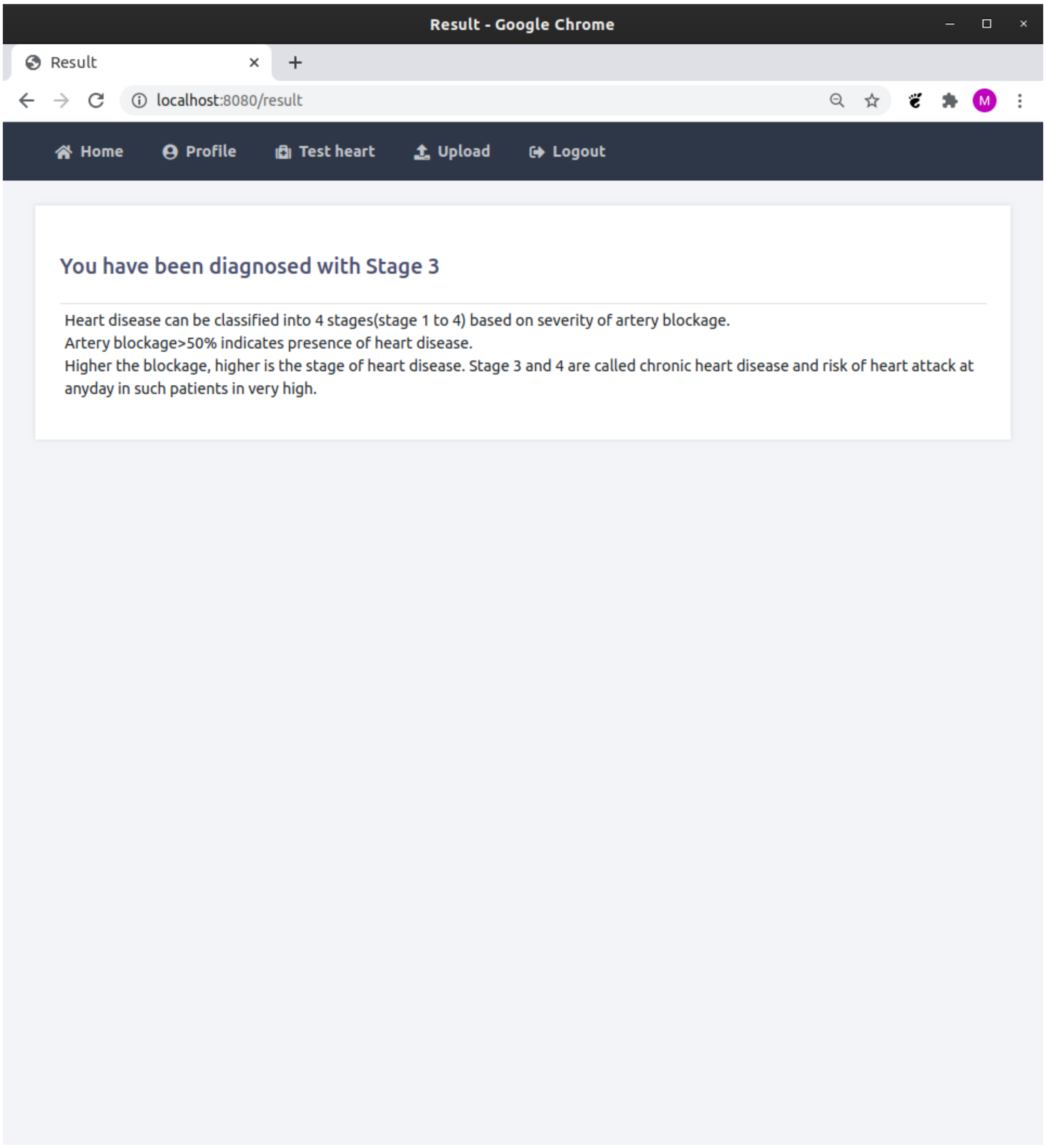
Username: Mrinalini

Email: abc@gmail.com

5.5 Prediction if risk of heart disease

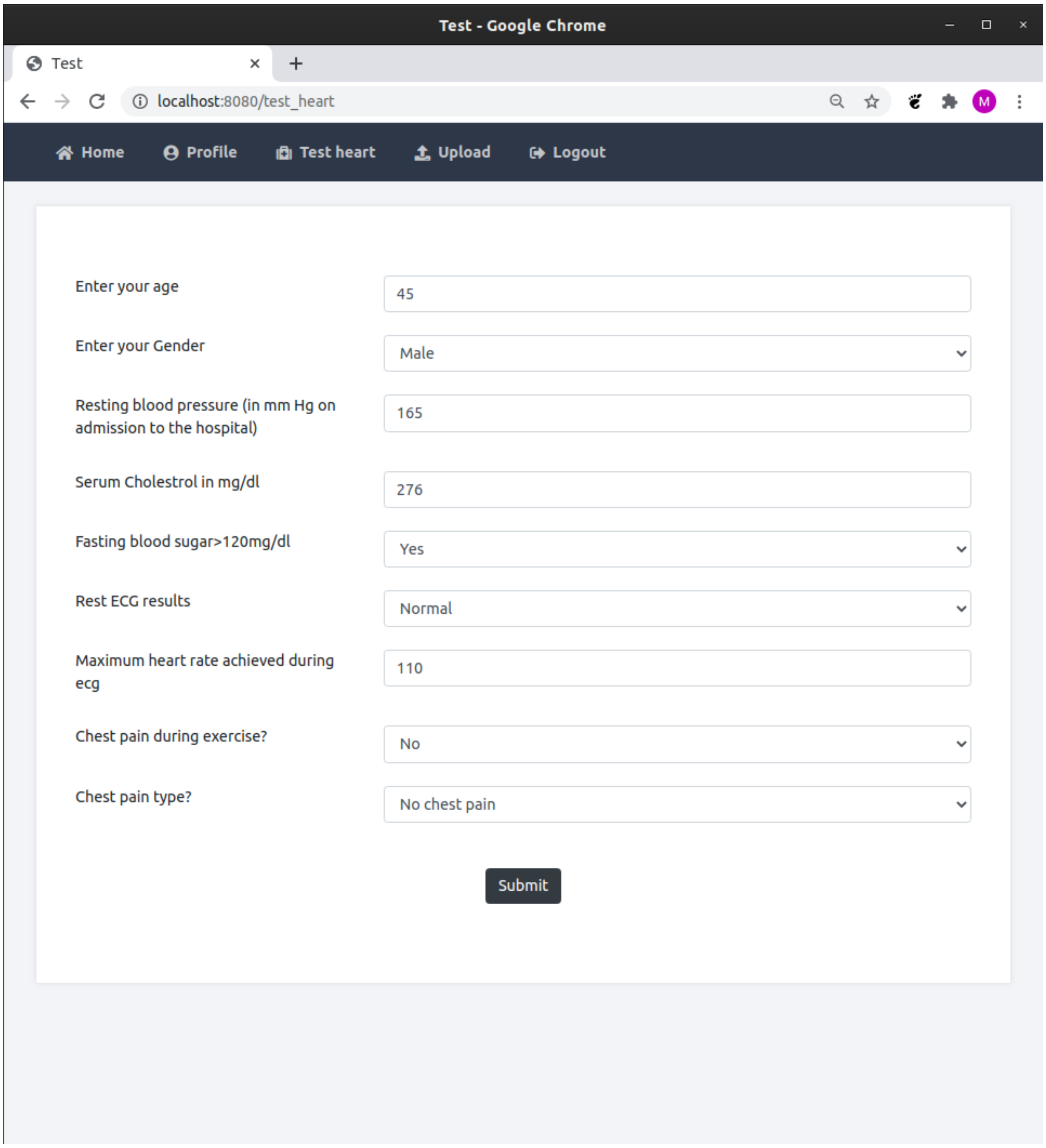


Enter your age	67
Enter your Gender	Female
Resting blood pressure (in mm Hg on admission to the hospital)	160
Serum Cholestrol in mg/dl	286
Fasting blood sugar>120mg/dl	No
Rest ECG results	Showing probable or definite left ventricular hypertrophy by Estes' criteria
Maximum heart rate achieved during ecg	129
Chest pain during exercise?	Yes
Chest pain type?	Asymptomatic chest pain(Select if the above are not applicable)

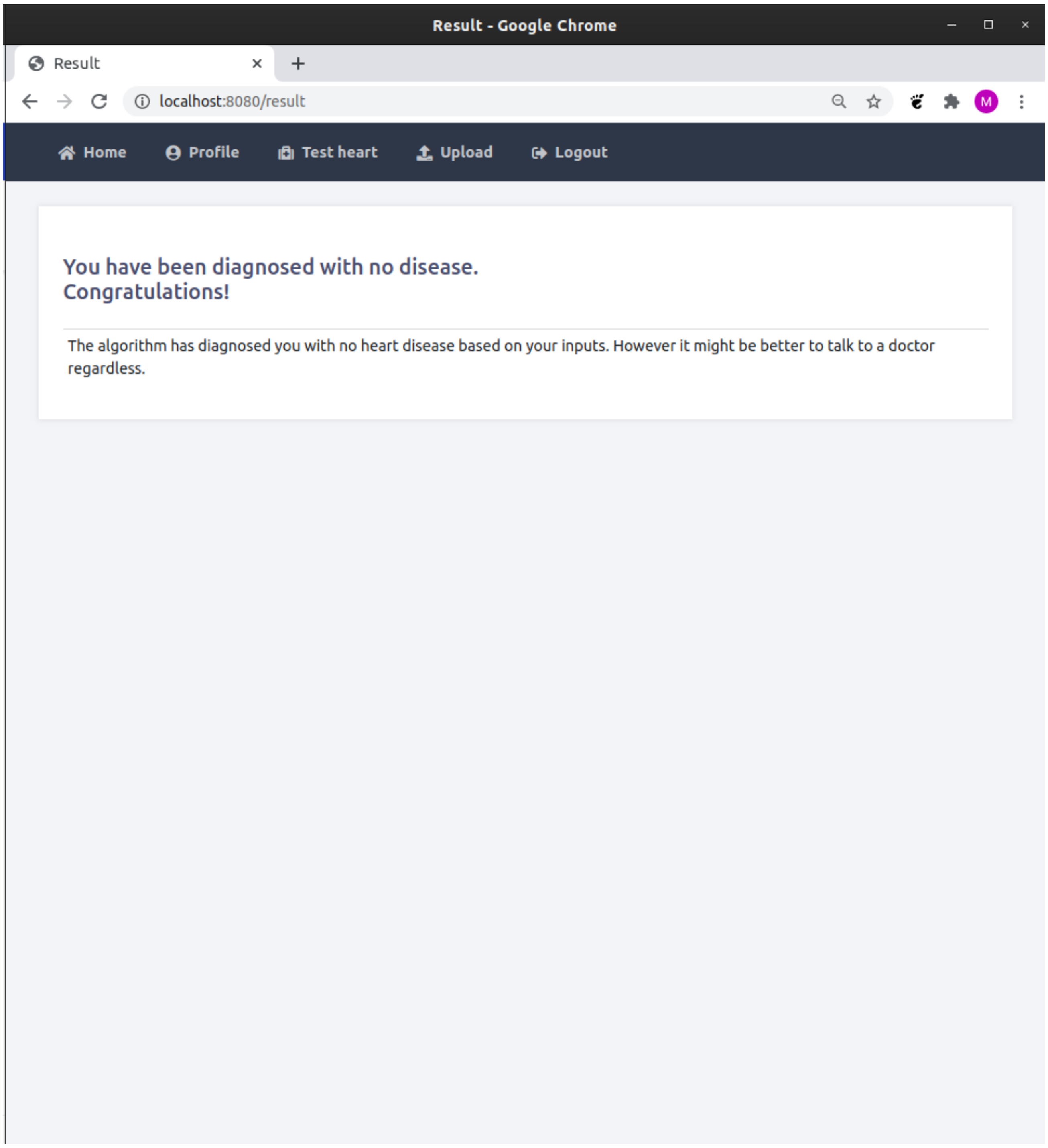


You have been diagnosed with Stage 3

Heart disease can be classified into 4 stages(stage 1 to 4) based on severity of artery blockage.
Artery blockage>50% indicates presence of heart disease.
Higher the blockage, higher is the stage of heart disease. Stage 3 and 4 are called chronic heart disease and risk of heart attack at anyday in such patients in very high.



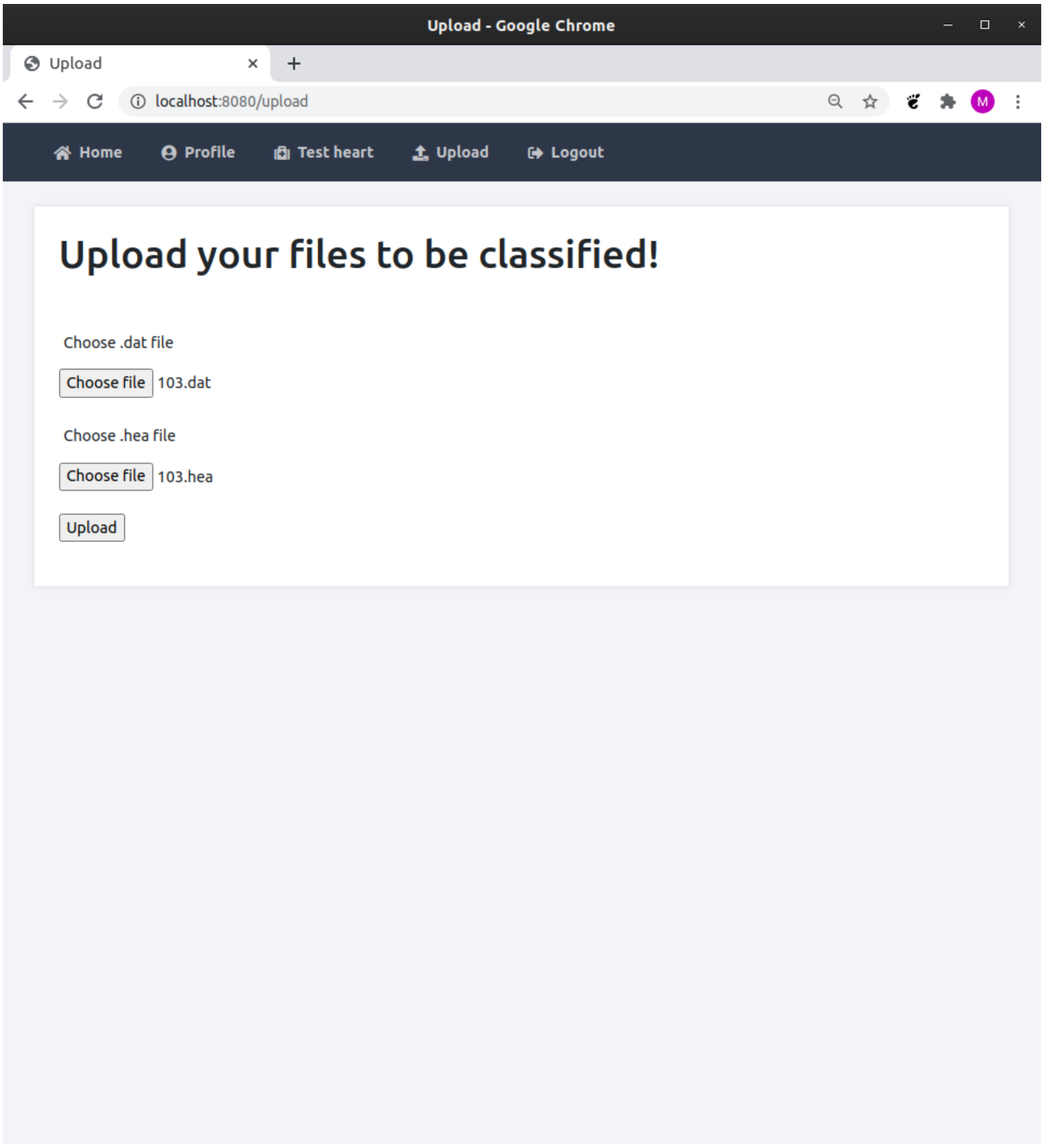
Enter your age	45
Enter your Gender	Male
Resting blood pressure (in mm Hg on admission to the hospital)	165
Serum Cholestrol in mg/dl	276
Fasting blood sugar>120mg/dl	Yes
Rest ECG results	Normal
Maximum heart rate achieved during ecg	110
Chest pain during exercise?	No
Chest pain type?	No chest pain



You have been diagnosed with no disease. Congratulations!

The algorithm has diagnosed you with no heart disease based on your inputs. However it might be better to talk to a doctor regardless.

5.6 Arrythmia prediction using signal file



Upload your files to be classified!

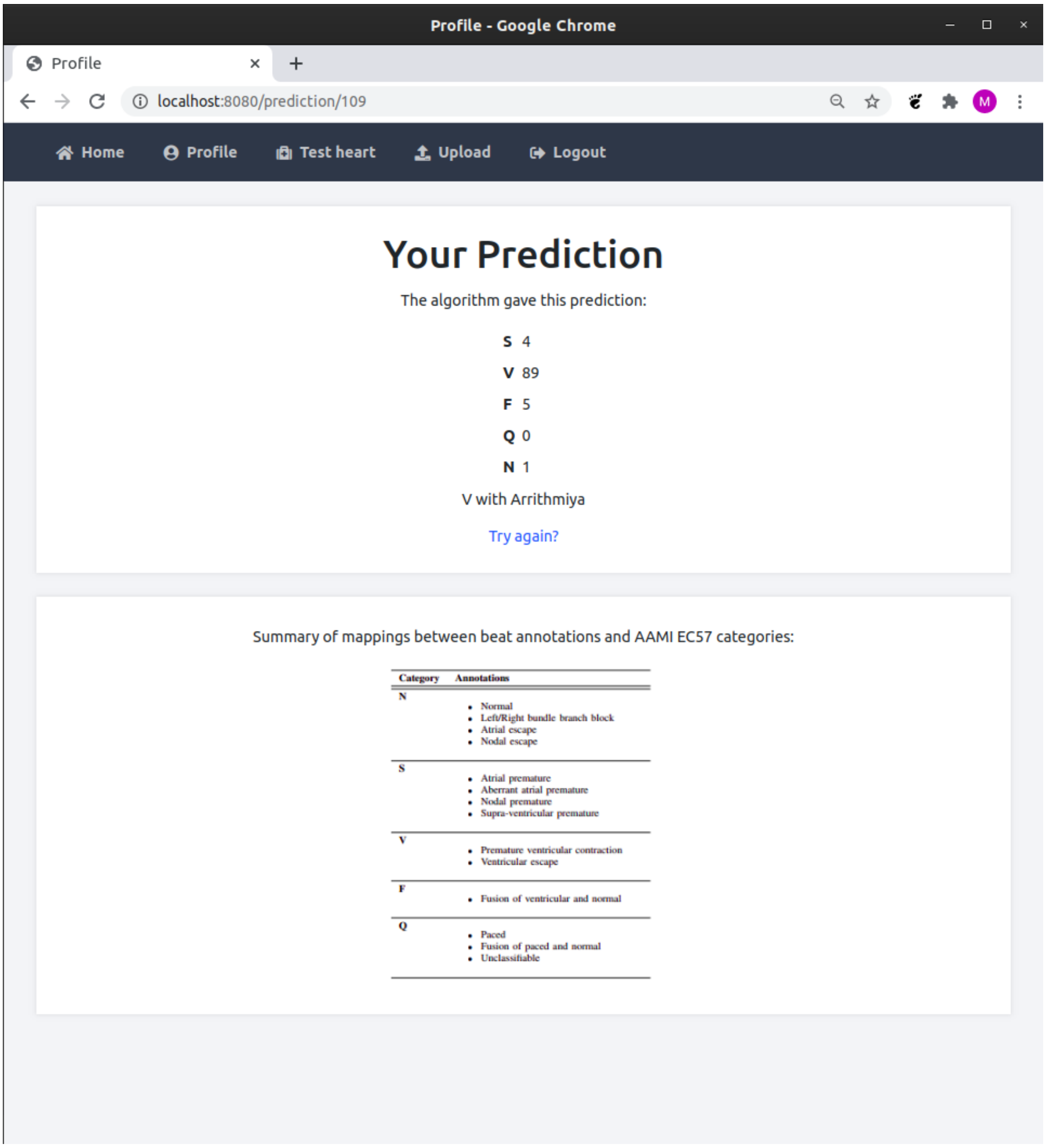
Choose .dat file

Choose file 103.dat

Choose .hea file

Choose file 103.hea

Upload



Your Prediction

The algorithm gave this prediction:

S	4
V	89
F	5
Q	0
N	1

V with Arrithmiya

Try again?

Summary of mappings between beat annotations and AAMI EC57 categories:

Category	Annotations
N	• Normal • Left/Right bundle branch block • Atrial escape • Nodal escape
S	• Atrial premature • Aberrant atrial premature • Nodal premature • Supra-ventricular premature
V	• Premature ventricular contraction • Ventricular escape
F	• Fusion of ventricular and normal
Q	• Paced • Fusion of paced and normal • Unclassifiable

6 Conclusion

The world population are looking for the healthy life throughout their life time. But, due to the changes in food habits and global warming in the environment creates some dangerous diseases in human beings. In that, most common one is heart diseases where everyone in the world scared about this and don't know when it will affect. In current scenario, electro cardiogram (ECG) is the easy and trustable method to identify the occurrence of heart diseases.

This project was successful in creating a user-friendly website that provides a platform to people to check about their heart conditions. They can do that in two ways firstly they can input their health information and find out how susceptible they are to heart diseases this was accomplished by using a random forest model on UCI dataset using supervised learning, another way is to input their ECG signal file which returns the annotations and arrythmia type for the person. Since in laboratories, arrhythmias are detected by continuous monitoring of the ECG signals which is very time consuming this could be a very useful technique.

In the future we can combine this project with IoT and attach sensors and Arduino/ Raspberry pie to get the signal values directly and give the predictions in real time.

References

[1] A. Subashini, L. SaiRamesh and G. Raghuraman, "Identification and Classification of Heart Beat by Analyzing ECG Signal using Naive Bayes," 2019 Third International Conference on Inventive Systems and Control (ICISC), Coimbatore, India, 2019, pp. 691-694, doi: 10.1109/ICISC44355.2019.9036455.

[2] Classification of Arrhythmia Using Machine Learning Techniques1,2 THARA SOMAN PATRICK O. BOBBIE

[3] Heart diseases prediction based on ECG signals' classification using a genetic-fuzzy system and dynamical model of ECG signals. Crossref DOI link: https://doi.org/10.1016/J.BSPC.2014.08.010. Published: 2014-11

[4] Classification of ECG Arrhythmia using Recurrent Neural Networks. Crossref DOI link: https://doi.org/10.1016/j.procs.2018.05.045. Published: 2018

[5] Shah, D., Patel, S. & Bharti, S.K. Heart Disease Prediction using Machine Learning Techniques. SN COMPUT. SCI. 1, 345 (2020). https://doi.org/10.1007/s42979-020-00365-y

[6] Heart Disease Prediction Using CNN Algorithm https://link.springer.com/article/10.1007/s42979-020-0097-6

[7] "Arrhythmia detection and classification using morphological and dynamic features of ECG signals' by Can Ye 1, Miguel Tavares Coimbra, B K Vijaya Kumar https://pubmed.ncbi.nlm.nih.gov/21097000/

[8] "Arrhythmia Classification of ECG Signals Using Hybrid Features" by Syed Muhammad Anwar, Maheen Gul, Muhammad Majid and Majdi Alnowami Published: 2018

[9] Rajendra Acharya - "Application of deep learning techniques for heartbeats detection using ECG signals-analysis and review" https://Sciencedirect.com/science/article/abs/pii/S0010482520301104

[10] Çınar A Tuncer- "Classification of Heart Diseases Based on ECG Signals Using Long Short-Term Memory" Published: 2019 https://pubmed.ncbi.nlm.nih.gov/30440962/