# **Artificial Neural Network: Project Report**

# **RA1811032010005** Aryamaan Parida **RA1811032010004** Jitin Goyal

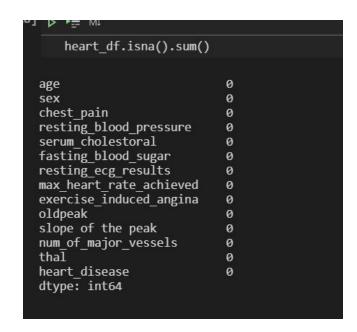
# **Objectives:**

- To create an Artificial Neural Network that predicts whether a person has heart disease or not.
- To understand/gain knowledge about how an actual Neural Network operates and its implementation.

## **Preprocessing performed to raw data:**

 Since a Neural Network expects data to be numeric in nature and not contain missing values we used two functions to verify the same i.e. "heart\_df.isna().sum()" and "heart\_df.dtypes"

```
heart df.dtypes
                            float64
age
                            float64
sex
chest pain
                            float64
resting blood pressure
                            float64
serum cholestoral
                            float64
fasting blood sugar
                            float64
resting ecg results
                            float64
max heart rate achieved
                            float64
exercise induced angina
                            float64
                            float64
oldpeak
slope of the peak
                            float64
num of major vessels
                            float64
thal
                            float64
heart disease
                              int64
dtype: object
```



• Finally, to perform dot products we **reshaped the y-label to a 1-D** array. We also **standardised the dataset using the Standard Scaler** module of sklearn.

#### **Features extracted:**

We did not perform any kind of Feature Extraction on the model, the features were kept the same. The dataset gave an output of either 1 or 2 depending on whether the person has Heart Disease or not. For our convenience we changed this to 0 or 1. We also dropped the Heart Disease variable because this is our target variable. As mentioned above, the dataset has no missing values and therefore further cleaning was not required.

```
#replace target class with 0 and 1
#1 means "have heart disease" and 0 means "do not have heart disease"
heart_df['heart_disease'] = heart_df['heart_disease'].replace(1, 0)
heart_df['heart_disease'] = heart_df['heart_disease'].replace(2, 1)
```

## **Models used:**

A Feedforward Neural Network has been used as shown below. Excluding the Heart Disease variable which was dropped we have 13 features, therefore the Input Layer has 13 nodes. The Hidden Layer has 8 nodes while the Output Layer has just 1 node as this is a Binary Classification task.

### **Details of Test and Train set:**

- Using the test\_train\_split from sklearn we split the data into Train Set and Test Set with the Test Set taking 20% of the data.
- The .shape function is used to display the Test Set and Train Set.

```
print(f"Shape of train set is {Xtrain.shape}")
  print(f"Shape of test set is {Xtest.shape}")
  print(f"Shape of train label is {ytrain.shape}")
  print(f"Shape of test labels is {ytest.shape}")

Shape of train set is (216, 13)
Shape of test set is (54, 13)
Shape of train label is (216, 1)
Shape of test labels is (54, 1)
```

## **Number of repetitions:** 100

# **Results:**

• The model depicts a **Train Accuracy** of **83.33** % and a **Test Accuracy of 85.18** %. We tried enhancing the **Learning Rate** and the **Number of Repetitions** however this led to a decrease in Test Accuracy and an increase in Train Accuracy.

- Acknowledging the importance of the Test Accuracy and to avoid overfitting we decided to keep the values unchanged.
- As shown below, the loss curve depicts a decrease in loss over time as the number of iterations increase (it nearly approaches zero).
- Considering the fact this was our first attempt to build a Neural Network from scratch we view these to be great accuracies.

```
train_pred = nn.predict(Xtrain)
test_pred = nn.predict(Xtest)

print("Train accuracy is {}".format(nn.acc(ytrain, train_pred)))
print("Test accuracy is {}".format(nn.acc(ytest, test_pred)))

Train accuracy is 83.333333333333334
Test accuracy is 85.18518518519
```

