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
```

```
age                        float64
sex                        float64
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
oldpeak                    float64
slope of the peak          float64
num_of_major_vessels       float64
thal                       float64
heart_disease                int64
dtype: object
```

```
heart_df.isna().sum()
```

```
age                        0
sex                        0
chest_pain                 0
resting_blood_pressure     0
serum_cholestoral          0
fasting_blood_sugar        0
resting_ecg_results        0
max_heart_rate_achieved    0
exercise_induced_angina    0
oldpeak                    0
slope of the peak          0
num_of_major_vessels       0
thal                       0
heart_disease              0
dtype: int64
```

- 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.

```
def forward_propagation(self):
    '''
    Performs the forward propagation
    '''

    Z1 = self.X.dot(self.params['W1']) + self.params['b1']
    A1 = self.relu(Z1)
    Z2 = A1.dot(self.params['W2']) + self.params['b2']
    yhat = self.sigmoid(Z2)
    loss = self.entropy_loss(self.y,yhat)

    # save calculated parameters
    self.params['Z1'] = Z1
    self.params['Z2'] = Z2
    self.params['A1'] = A1
```

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 83.33333333333334
Test accuracy is 85.18518518518519
```

```
nn.plot_loss()
```