Date: **Ex No:** 10	**Title of the Lab** Training of ANN using Deep Learning for Human Attack Prediction Dataset	**Name:** Yuvraj Singh Chauhan **Registration Number:** RA1911027010058 **Section:** N1 **Lab Batch:** 1 **Day Order:** 3

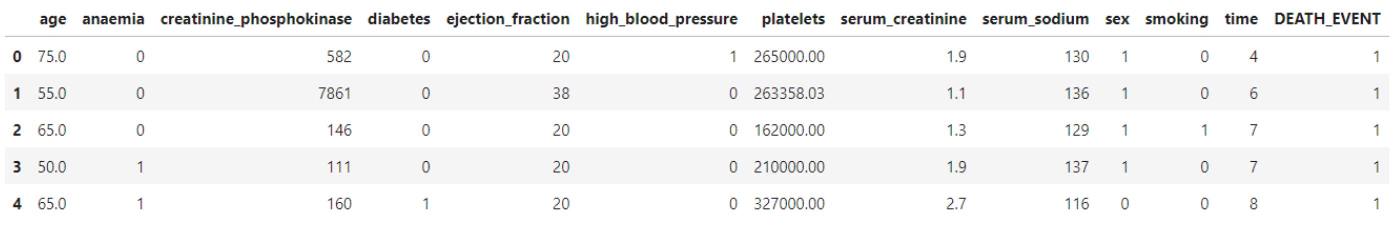
Aim:

To train an ANN for Human Attack Prediction dataset.

Description of Human Attack Prediction:

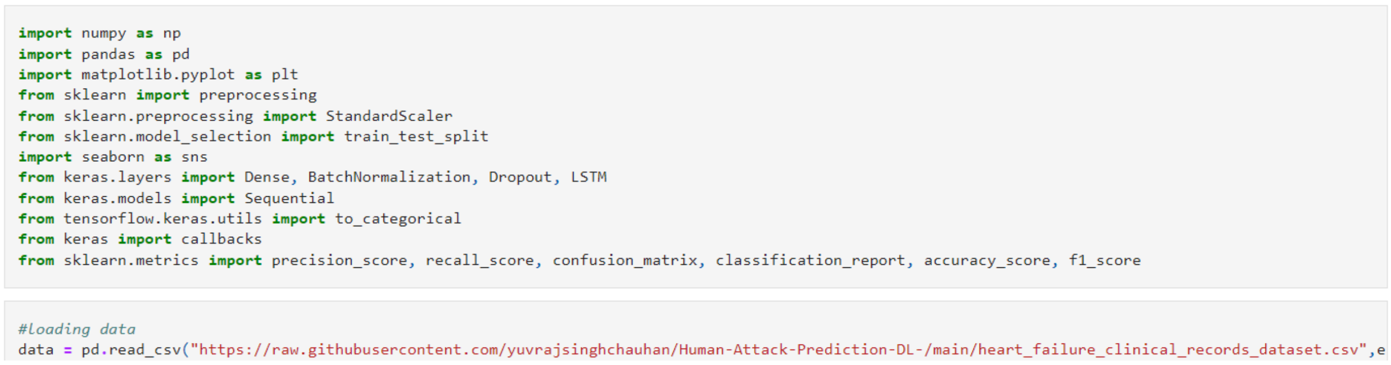
Nowadays, health diseases are increasing day by day due to life style, hereditary. Especially, heart disease has become more common these days. Each individual has different values for Blood pressure, cholesterol and pulse rate. In health care system there is large amount of data but poor in knowledge. The reason behind it is lack of effective analysis system to discover hidden relationship and data. Because of these data mining is used for the extracting and analyzing the knowledge. Numbers of data mining techniques are used like decision tree, Naïve Bayes, KNN, K-means, BP algorithm. But neural network is one of the most important techniques in data mining. We use the Intelligent Algorithm i.e., ANN (Artificial Neural Network). By using data mining techniques, it takes less time for the prediction of the disease with more accuracy.

Dataset:



	age	anaemia	creatinine_phosphokinase	diabetes	ejection_fraction	high_blood_pressure	platelets	serum_creatinine	serum_sodium	sex	smoking	time	DEATH_EVENT
0	75.0	0	582	0	20	1	265000.00	1.9	130	1	0	4	1
1	55.0	0	7861	0	38	0	263358.03	1.1	136	1	0	6	1
2	65.0	0	146	0	20	0	162000.00	1.3	129	1	1	7	1
3	50.0	1	111	0	20	0	210000.00	1.9	137	1	0	7	1
4	65.0	1	160	1	20	0	327000.00	2.7	116	0	0	8	1

Importing Files for DL:

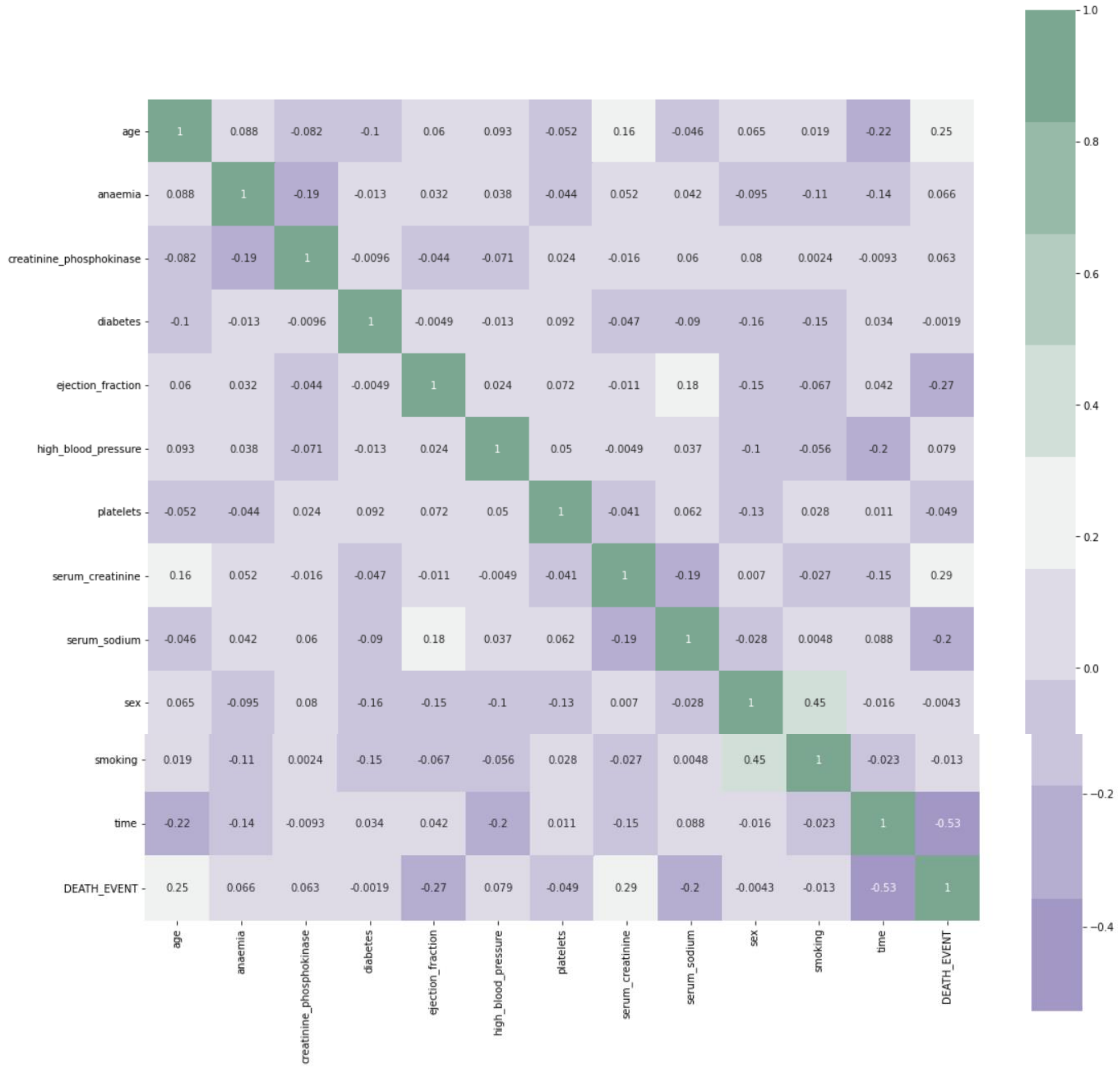


```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn import preprocessing
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
import seaborn as sns
from keras.layers import Dense, BatchNormalization, Dropout, LSTM
from keras.models import Sequential
from tensorflow.keras.utils import to_categorical
from keras import callbacks
from sklearn.metrics import precision_score, recall_score, confusion_matrix, classification_report, accuracy_score, f1_score
```

```
#loading data
data = pd.read_csv("https://raw.githubusercontent.com/yuvrajsinghchauhan/Human-Attack-Prediction-DL-/main/heart_failure_clinical_records_dataset.csv",e
```

Implementing Corelation Matrix:

```
#Examaning a corelation matrix of all the features
cmap = sns.diverging_palette(275,150,  s=40, l=65, n=9)
corrmat = data.corr()
plt.subplots(figsize=(18,18))
sns.heatmap(corrmat,cmap= cmap,annot=True, square=True);
```

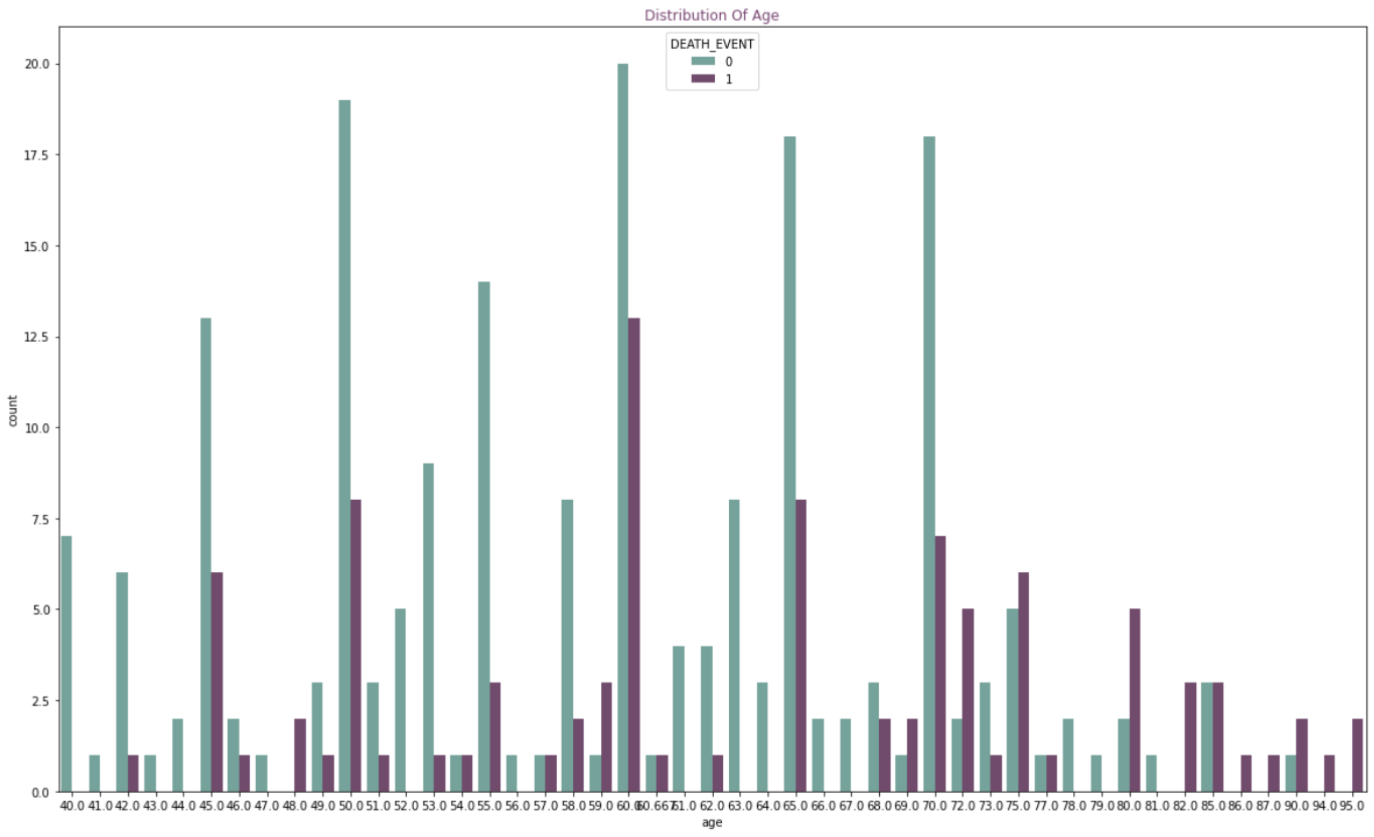


	age	anaemia	creatinine_phosphokinase	diabetes	ejection_fraction	high_blood_pressure	platelets	serum_creatinine	serum_sodium	sex	smoking	time	DEATH_EVENT
age	1	0.088	-0.082	-0.1	0.06	0.093	-0.052	0.16	-0.046	0.065	0.019	-0.22	0.25
anaemia	0.088	1	-0.19	-0.013	0.032	0.038	-0.044	0.052	0.042	-0.095	-0.11	-0.14	0.066
creatinine_phosphokinase	-0.082	-0.19	1	-0.0096	-0.044	-0.071	0.024	-0.016	0.06	0.08	0.0024	-0.0093	0.063
diabetes	-0.1	-0.013	-0.0096	1	-0.0049	-0.013	0.092	-0.047	-0.09	-0.16	-0.15	0.034	-0.0019
ejection_fraction	0.06	0.032	-0.044	-0.0049	1	0.024	0.072	-0.011	0.18	-0.15	-0.067	0.042	-0.27
high_blood_pressure	0.093	0.038	-0.071	-0.013	0.024	1	0.05	-0.0049	0.037	-0.1	-0.056	-0.2	0.079
platelets	-0.052	-0.044	0.024	0.092	0.072	0.05	1	-0.041	0.062	-0.13	0.028	0.011	-0.049
serum_creatinine	0.16	0.052	-0.016	-0.047	-0.011	-0.0049	-0.041	1	-0.19	0.007	-0.027	-0.15	0.29
serum_sodium	-0.046	0.042	0.06	-0.09	0.18	0.037	0.062	-0.19	1	-0.028	0.0048	0.088	-0.2
sex	0.065	-0.095	0.08	-0.16	-0.15	-0.1	-0.13	0.007	-0.028	1	0.45	-0.016	-0.0043
smoking	0.019	-0.11	0.0024	-0.15	-0.067	-0.056	0.028	-0.027	0.0048	0.45	1	-0.023	-0.013
time	-0.22	-0.14	-0.0093	0.034	0.042	-0.2	0.011	-0.15	0.088	-0.016	-0.023	1	-0.53
DEATH_EVENT	0.25	0.066	0.063	-0.0019	-0.27	0.079	-0.049	0.29	-0.2	-0.0043	-0.013	-0.53	1

Evaluating Age Distribution:

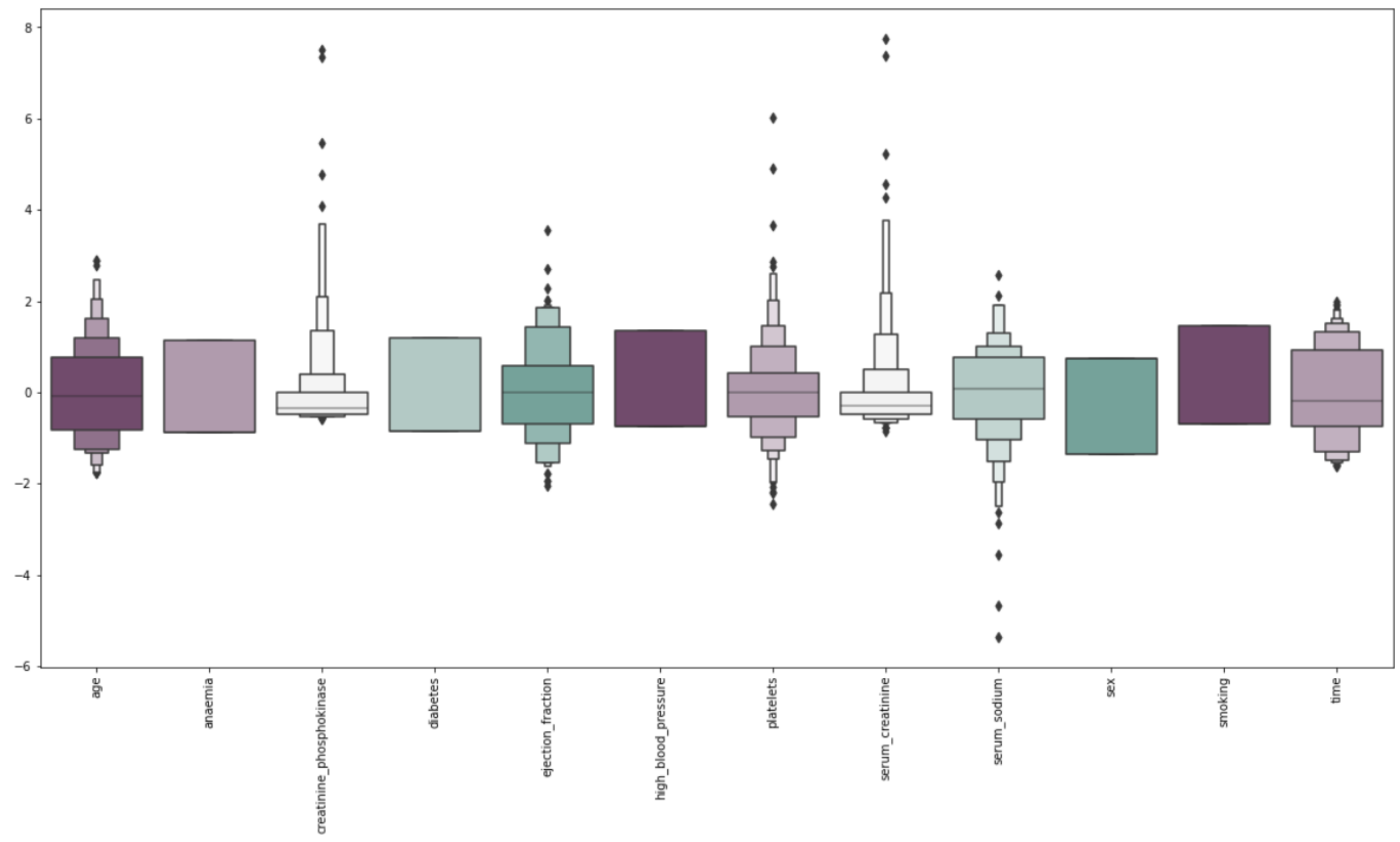
```
#Evauating age distrivution
plt.figure(figsize=(20,12))
#colours =["#774571","#b398af","#f1f1f1" ,"#afcdc7", "#6daa9f"]
Days_of_week=sns.countplot(x=data['age'],data=data, hue ="DEATH_EVENT",palette = cols)
Days_of_week.set_title("Distribution Of Age", color="#774571")
```

```
Text(0.5, 1.0, 'Distribution Of Age')
```



Scaled Features:

```
#looking at the scaled features
colours =["#774571","#b398af","#f1f1f1" ,"#afcdc7", "#6daa9f"]
plt.figure(figsize=(20,10))
sns.boxenplot(data = X_df,palette = colours)
plt.xticks(rotation=90)
plt.show()
```



Developing The ANN:

```
#spliting test and training sets
X_train, X_test, y_train,y_test = train_test_split(X_df,y,test_size=0.25,random_state=7)
```

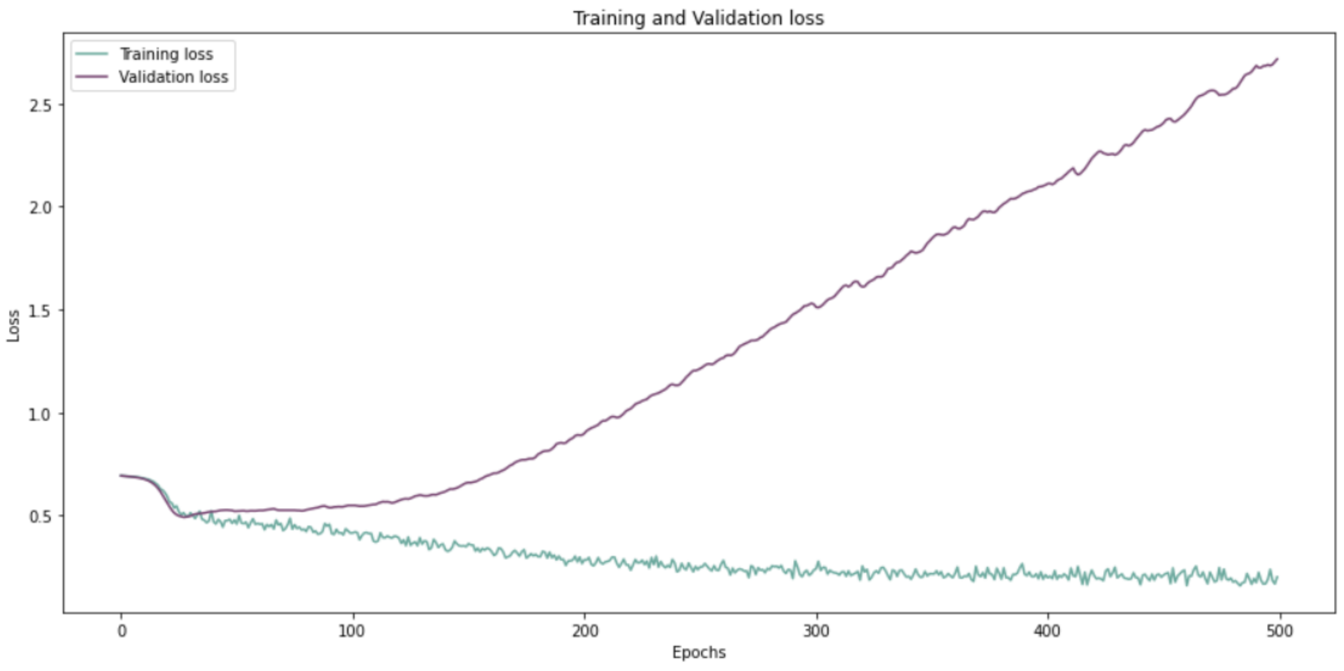
```
early_stopping = callbacks.EarlyStopping(
    min_delta=0.001, # minimium amount of change to count as an improvement
    patience=20, # how many epochs to wait before stopping
    restore_best_weights=True)

# Initialising the NN
model = Sequential()

# layers
model.add(Dense(units = 16, kernel_initializer = 'uniform', activation = 'relu', input_dim = 12))
model.add(Dense(units = 8, kernel_initializer = 'uniform', activation = 'relu'))
model.add(Dropout(0.25))
model.add(Dense(units = 4, kernel_initializer = 'uniform', activation = 'relu'))
model.add(Dropout(0.5))
model.add(Dense(units = 1, kernel_initializer = 'uniform', activation = 'sigmoid'))
from tensorflow.keras.optimizers import SGD
# Compiling the ANN
model.compile(optimizer = 'adam', loss = 'binary_crossentropy', metrics = ['accuracy'])

# Train the ANN
history = model.fit(X_train, y_train, batch_size = 32, epochs = 500, validation_split=0.2)
```

```
Epoch 1/500
6/6 [==============================] - 1s 40ms/step - loss: 0.6928 - accuracy: 0.6480 - val_loss: 0.6922 - val_accuracy: 0.6667
Epoch 2/500
6/6 [==============================] - 0s 6ms/step - loss: 0.6919 - accuracy: 0.6480 - val_loss: 0.6911 - val_accuracy: 0.6667
Epoch 3/500
6/6 [==============================] - 0s 6ms/step - loss: 0.6909 - accuracy: 0.6480 - val_loss: 0.6901 - val_accuracy: 0.6667
Epoch 4/500
```

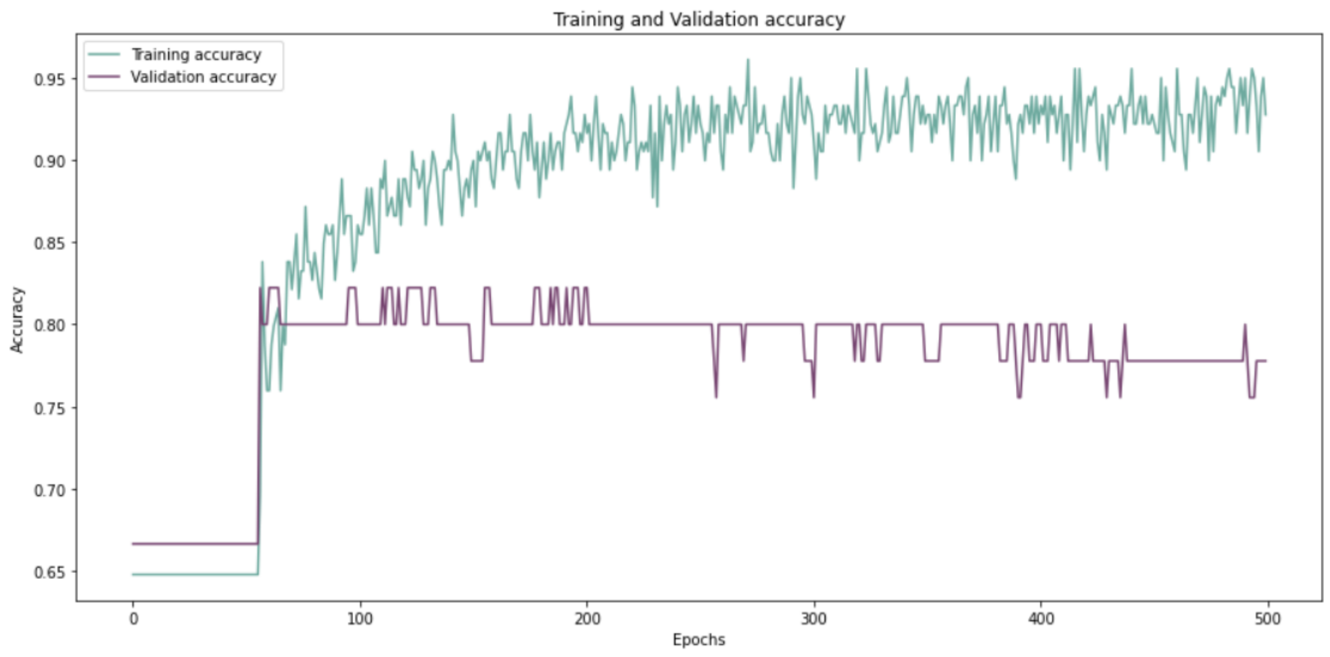
```
val_accuracy = np.mean(history.history['val_accuracy'])
print("\n%s: %.2f%%" % ('val_accuracy', val_accuracy*100))
```

```
val_accuracy: 78.08%
```

```
history_df = pd.DataFrame(history.history)
plt.figure(figsize=(15,7))
plt.plot(history_df.loc[:, ['loss']], "#6daa9f", label='Training loss')
plt.plot(history_df.loc[:, ['val_loss']],"#774571", label='Validation loss')
plt.title('Training and Validation loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend(loc="best")

plt.show()
```



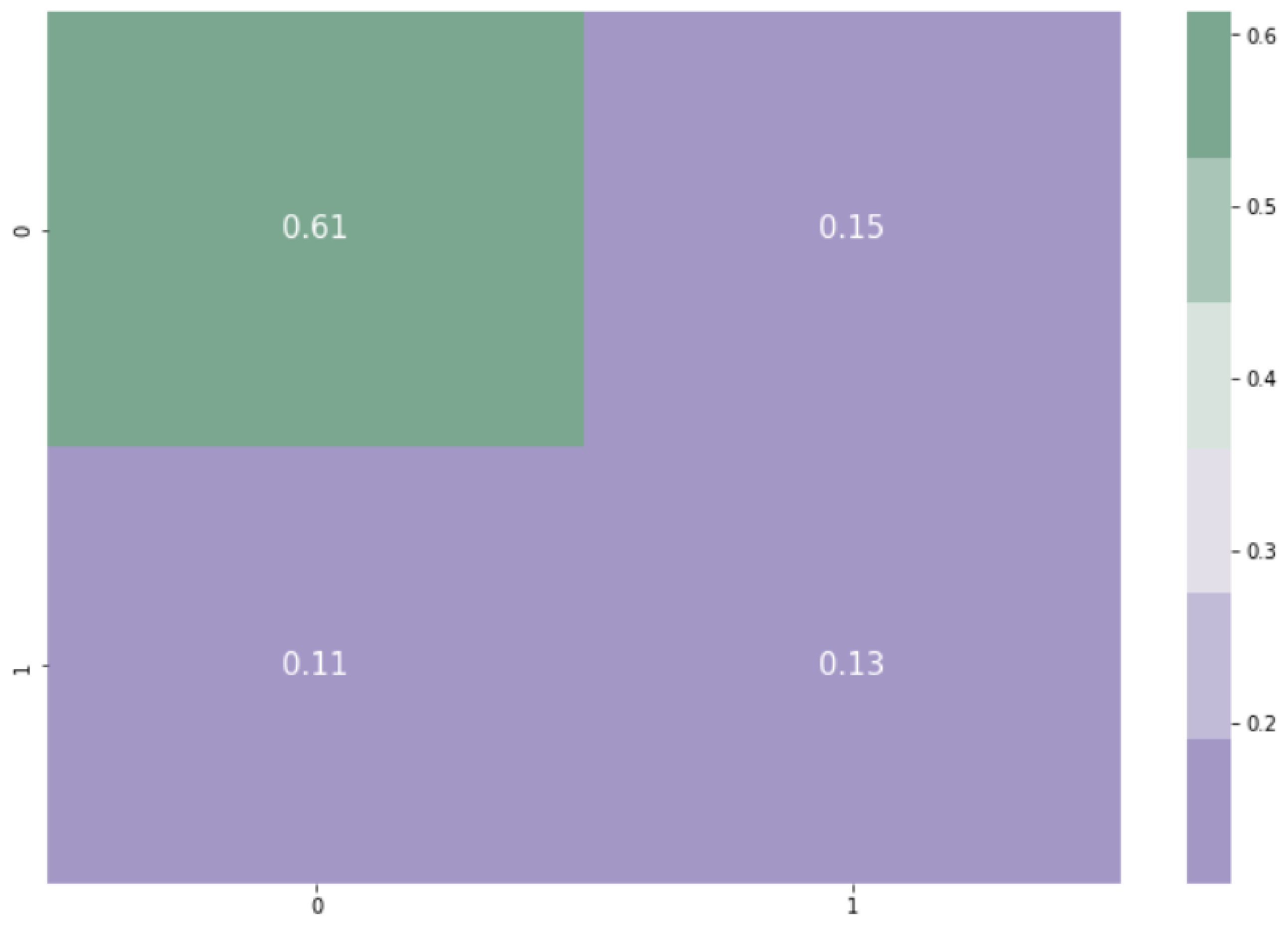
```
history_df = pd.DataFrame(history.history)
plt.figure(figsize=(15,7))
plt.plot(history_df.loc[:, ['accuracy']], "#6daa9f", label='Training accuracy')
plt.plot(history_df.loc[:, ['val_accuracy']], "#774571", label='Validation accuracy')

plt.title('Training and Validation accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
```



Predicting the Test Set Result:

```
# Predicting the test set results
y_pred = model.predict(X_test)
y_pred = (y_pred > 0.5)
np.set_printoptions()
# confusion matrix
cmap1 = sns.diverging_palette(275,150,  s=40, l=65, n=6)
plt.subplots(figsize=(12,8))
cf_matrix = confusion_matrix(y_test, y_pred)
sns.heatmap(cf_matrix/np.sum(cf_matrix), cmap = cmap1, annot = True, annot_kws = {'size':15})
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f0da8804390>
```



Conclusion:

From the analysis it is concluded that artificial neural network algorithm is best for classification of knowledge data from large amount of medical data. It has good performance with increase in efficiency with an accuracy of 78% when provided with normalized data. The data is normalized using a classifier. The Artificial neural network is one of the best for heart disease prediction.

Signature of the Student

[YUVRAJ SINGH CHAUHAN]