Machine Learning Assignment

Name :Muhammad Hassan Reg No:FA21-BCE-051

Assignment :03 Submitted to:Dr Shoaib Azmat

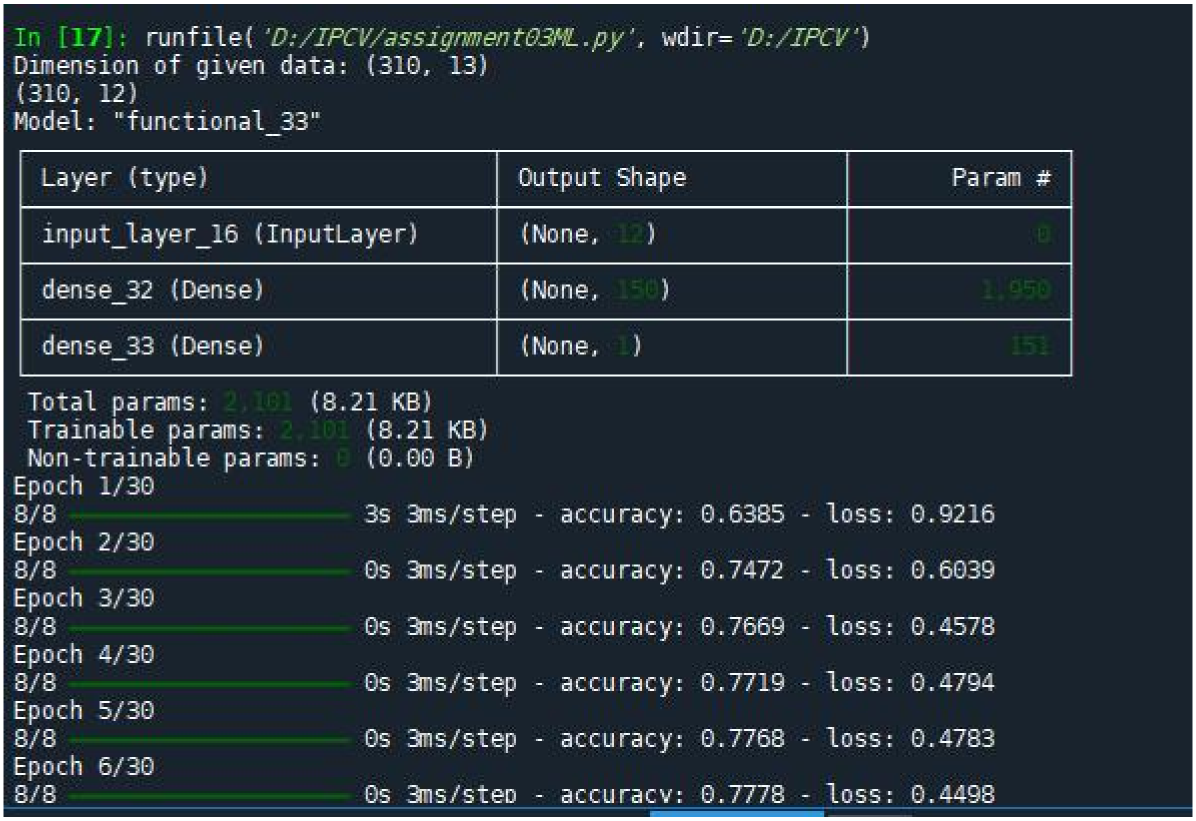
Task 01 :

Code:

```
import pandas as pd
import numpy as np
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix, accuracy_score
from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.keras.layers import Rescaling, Normalization
from sklearn.exceptions import ConvergenceWarning
from sklearn.neural_network import MLPClassifier
#help in not showing warning b/c warning make hurdel in output analysis
import warnings
warnings.filterwarnings("ignore", category=ConvergenceWarning)
# Read and show data
data = pd.read_csv('C:\\Users\\butt0\\Downloads\\assignemnt 02ML\\spine.csv')
print('Dimension of given data:', data.shape)
#extract features and split in 80% train and 20% test set
x = data[['Col1', 'Col2', 'Col3', 'Col4', 'Col5', 'Col6', 'Col7', 'Col8', 'Col9', 'Col10', 'Col11', 'Col12']].values
y = data['Class_label'].values
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=104)
print(x.shape)
# Build the model using the functional API
input_layer = keras.Input(shape=(12,))
#x = Rescaling(scale=1.0 /310)(x)
dense_layer = layers.Dense(150, activation='relu')(input_layer)
#dense_layer = layers.BatchNormalization()(input_layer)
output_layer = layers.Dense(1, activation='sigmoid')(dense_layer)
model = keras.Model(input_layer,output_layer)
# Summarize the model
model.summary()
# Compile the model
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
# Train the model
model.fit(x_train, y_train, epochs=30)
# Evaluate the model
loss, accuracy = model.evaluate(x_test, y_test)
print(f'Test Accuracy of traning NN: {accuracy}')
# Predictions
y_pred = (model.predict(x_test) > 0.5).astype("int32")
#print(y_pred)
# Confusion matrix
cm = confusion_matrix(y_test, y_pred)
print('Confusion Matrix:')
print(cm)
# Accuracy
acc = accuracy_score(y_test, y_pred)
print(f'Accuracy of Testing NN: {acc}')
linear = LogisticRegression()
linear.fit(x_train,y_train)
```

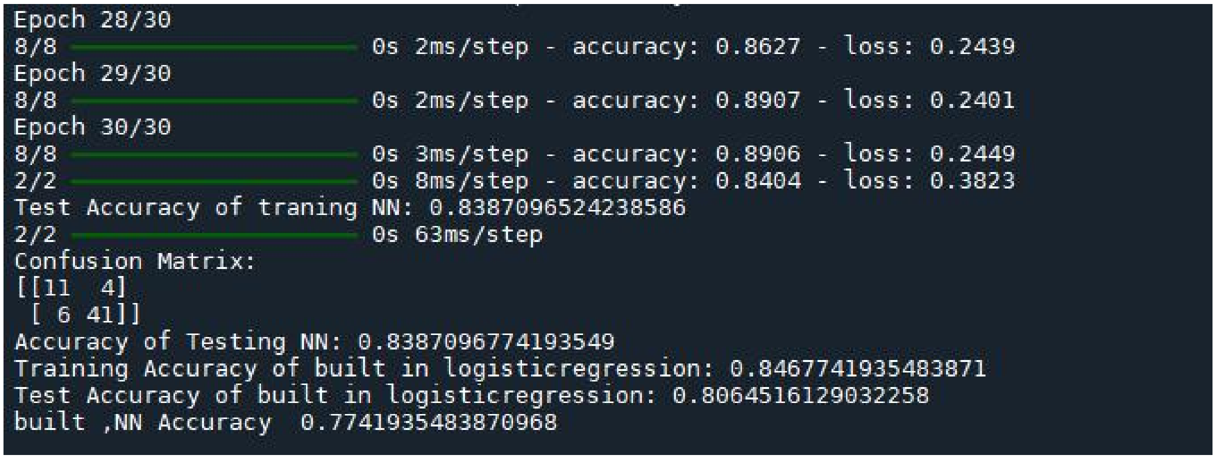
```
accuracy_train = accuracy_score(y_train, linear.predict(x_train))
accuracy_test = accuracy_score(y_test,linear.predict(x_test))
print("Training Accuracy of built in logisticregression:", accuracy_train)
print("Test Accuracy of built in logisticregression:", accuracy_test)
mlp = MLPClassifier(hidden_layer_sizes=(1))
mlp.fit(x_train, y_train.ravel())
print("built ,NN Accuracy ", accuracy_score(y_test,mlp.predict(x_test)))
```

Result:



```
In [17]: runfile('D:/IPCV/assignment03ML.py', wdir='D:/IPCV')
Dimension of given data: (310, 13)
(310, 12)
Model: "functional_33"
┌─────────────────────────────────┬────────────────────────┬───────────────┐
│ Layer (type)                    │ Output Shape           │       Param # │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ input_layer_16 (InputLayer)     │ (None, 12)             │             0 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ dense_32 (Dense)                │ (None, 150)            │         1,950 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ dense_33 (Dense)                │ (None, 1)              │           151 │
└─────────────────────────────────┴────────────────────────┴───────────────┘
 Total params: 2,101 (8.21 KB)
 Trainable params: 2,101 (8.21 KB)
 Non-trainable params: 0 (0.00 B)
Epoch 1/30
8/8 ━━━━━━━━━━━━━━━━━━━━ 3s 3ms/step - accuracy: 0.6385 - loss: 0.9216
Epoch 2/30
8/8 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.7472 - loss: 0.6039
Epoch 3/30
8/8 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.7669 - loss: 0.4578
Epoch 4/30
8/8 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.7719 - loss: 0.4794
Epoch 5/30
8/8 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.7768 - loss: 0.4783
Epoch 6/30
8/8 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.7778 - loss: 0.4498
```

```
.                      .                                         .
.                      .                                         .
.                      .                                         .
.                      .                                         .
.                      .                                         .
```



```
Epoch 28/30
8/8 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.8627 - loss: 0.2439
Epoch 29/30
8/8 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.8907 - loss: 0.2401
Epoch 30/30
8/8 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.8906 - loss: 0.2449
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - accuracy: 0.8404 - loss: 0.3823
Test Accuracy of traning NN: 0.8387096524238586
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 63ms/step
Confusion Matrix:
[[11  4]
 [ 6 41]]
Accuracy of Testing NN: 0.8387096774193549
Training Accuracy of built in logisticregression: 0.8467741935483871
Test Accuracy of built in logisticregression: 0.8064516129032258
built ,NN Accuracy  0.7741935483870968
```

Response/Comments:

Our NN model predict test data of 83% and training accuracy is 84% which show that our mode make appropriate fitting ,if we compare with previously implementation ,their is case over fitting.
Difference in built_in and our model is due to noise,we handle this issues by using ResNET,

In this model ,we insert hidden layer (dense layer) of 150 neuron , we are tackling only one class not a multi class so we make output dense layer with one neuron which predict 0 or 1 (normal or abnormal)
if we compare NN with previous algorithm it is easy to implement ,learn feature well ,not make complex (overfitting or underfitting mostly),its hidden layer neuron learn complex feature with itself.
And we also see that logistic regression algorithm make model ourfitt because it traning and testing accuracy are different