Name: Natiola, Henry Jay P.

Course and Section: CPE 019 - CPE32S9

Date of Submission: 03/20/24

Instructor: Engr. Roman Richard

In this assignment, you are task to build a multilayer perceptron model. The following are the requirements:

Choose anu dataset

Explain the problem you are trying to solve

Create your own model

Evaluate the accuracy of your model

```
# Import needed Libraries
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
```

DATASET - INJURY DATASET

```
# Load the dataset
url = "https://raw.githubusercontent.com/Jayryy/CPE-019-CPE32S9/main/Dataset/injury_data.csv"
data = pd.read_csv(url)
```

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 7 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   Player_Age           1000 non-null   int64
 1   Player_Weight        1000 non-null   float64
 2   Player_Height        1000 non-null   float64
 3   Previous_Injuries    1000 non-null   int64
 4   Training_Intensity   1000 non-null   float64
 5   Recovery_Time        1000 non-null   int64
 6   Likelihood_of_Injury  1000 non-null   int64
dtypes: float64(3), int64(4)
memory usage: 54.8 KB
```

```
data.head(10)
```

	Player_Age	Player_Weight	Player_Height	Previous_Injuries	Training_Intensity	Rec
0	24	66.251933	175.732429	1	0.457929	
1	37	70.996271	174.581650	0	0.226522	
2	32	80.093781	186.329618	0	0.613970	
3	28	87.473271	175.504240	1	0.252858	
4	25	84.659220	190.175012	0	0.577632	
5	38	75.820549	206.631824	1	0.359209	
6	24	70.126050	177.044588	0	0.823552	
7	36	79.038206	181.523155	1	0.820696	
8	28	64.086096	183.794821	1	0.477350	
9	28	66.829986	198.115048	1	0.350819	

Next steps: View recommended plots

Explain the problem you are trying to solve

The problem being solve of this dataset is to predict the possible injury of the player that participated in the dataset. So by applying multilayer perceptron we can analyze each factory to predict the occurance of injury on the players. So basically by applying a multilayer perceptron to analyze the various factors of the injury dataset, we can effectively predict the likelihood of player injuries.

Create your own model

```
data.head(20)
```

	Player_Age	Player_Weight	Player_Height	Previous_Injuries	Training_Intensity	Re
0	24	66.251933	175.732429	1	0.457929	
1	37	70.996271	174.581650	0	0.226522	
2	32	80.093781	186.329618	0	0.613970	
3	28	87.473271	175.504240	1	0.252858	
4	25	84.659220	190.175012	0	0.577632	
5	38	75.820549	206.631824	1	0.359209	
6	24	70.126050	177.044588	0	0.823552	
7	36	79.038206	181.523155	1	0.820696	
8	28	64.086096	183.794821	1	0.477350	
9	28	66.829986	198.115048	1	0.350819	
10	38	90.097713	179.173522	0	0.362560	
11	21	79.020345	171.709831	0	0.805715	
12	25	71.524733	167.336595	1	0.328179	
13	20	83.291387	179.420161	0	0.209059	
14	39	87.869750	175.516198	1	0.084688	
15	38	70.553624	200.100093	1	0.466693	
16	19	70.725921	197.698624	0	0.482132	
17	29	69.473542	169.656272	1	0.840909	
18	23	84.638503	172.892399	0	0.221239	
19	19	84.735574	180.737701	1	0.380991	

Next steps: View recommended plots

```
# Split the dataset into input features (X) and target variable (y)
X = data.drop('Likelihood_of_Injury', axis=1)
y = data['Likelihood_of_Injury']


# Split the data into training and validation sets
X_train, X_val, y_train, y_val = train_test_split(X, y, test_size=0.2, random_state=42)


# Standardize the input features
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_val = scaler.transform(X_val)


# Build the MLP model
model = Sequential()
model.add(Dense(32, activation='relu', input_shape=(X_train.shape[1],)))
model.add(Dense(16, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
```

```
# Compile the model
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])


# Train the model
model.fit(X_train, y_train, epochs=50, batch_size=32, verbose=1)
```

```
25/25 [==============================] - 0s 1ms/step - loss: 0.6831 - accuracy: 0.5738
Epoch 6/50
25/25 [==============================] - 0s 2ms/step - loss: 0.6810 - accuracy: 0.5638
Epoch 7/50
25/25 [==============================] - 0s 1ms/step - loss: 0.6797 - accuracy: 0.5650
Epoch 8/50
25/25 [==============================] - 0s 1ms/step - loss: 0.6780 - accuracy: 0.5725
Epoch 9/50
25/25 [==============================] - 0s 1ms/step - loss: 0.6766 - accuracy: 0.5725
Epoch 10/50
25/25 [==============================] - 0s 1ms/step - loss: 0.6754 - accuracy: 0.5775
Epoch 11/50
25/25 [==============================] - 0s 1ms/step - loss: 0.6746 - accuracy: 0.5813
Epoch 12/50
25/25 [==============================] - 0s 1ms/step - loss: 0.6727 - accuracy: 0.5800
Epoch 13/50
25/25 [==============================] - 0s 1ms/step - loss: 0.6710 - accuracy: 0.5875
Epoch 14/50
25/25 [==============================] - 0s 2ms/step - loss: 0.6701 - accuracy: 0.5875
Epoch 15/50
25/25 [==============================] - 0s 1ms/step - loss: 0.6684 - accuracy: 0.5888
Epoch 16/50
25/25 [==============================] - 0s 1ms/step - loss: 0.6669 - accuracy: 0.5913
Epoch 17/50
25/25 [==============================] - 0s 1ms/step - loss: 0.6659 - accuracy: 0.5888
Epoch 18/50
25/25 [==============================] - 0s 1ms/step - loss: 0.6648 - accuracy: 0.5950
Epoch 19/50
25/25 [==============================] - 0s 1ms/step - loss: 0.6637 - accuracy: 0.6012
Epoch 20/50
25/25 [==============================] - 0s 1ms/step - loss: 0.6626 - accuracy: 0.6000
Epoch 21/50
25/25 [==============================] - 0s 1ms/step - loss: 0.6612 - accuracy: 0.6025
Epoch 22/50
25/25 [==============================] - 0s 1ms/step - loss: 0.6607 - accuracy: 0.5938
Epoch 23/50
25/25 [==============================] - 0s 1ms/step - loss: 0.6595 - accuracy: 0.6012
Epoch 24/50
25/25 [==============================] - 0s 1ms/step - loss: 0.6579 - accuracy: 0.6087
Epoch 25/50
25/25 [==============================] - 0s 1ms/step - loss: 0.6562 - accuracy: 0.6075
Epoch 26/50
25/25 [==============================] - 0s 1ms/step - loss: 0.6551 - accuracy: 0.6087
Epoch 27/50
25/25 [==============================] - 0s 1ms/step - loss: 0.6550 - accuracy: 0.6125
Epoch 28/50
25/25 [==============================] - 0s 1ms/step - loss: 0.6524 - accuracy: 0.6087
Epoch 29/50
25/25 [==============================] - 0s 1ms/step - loss: 0.6515 - accuracy: 0.6187
Epoch 30/50
25/25 [==============================] - 0s 2ms/step - loss: 0.6496 - accuracy: 0.6225
Epoch 31/50
25/25 [==============================] - 0s 1ms/step - loss: 0.6485 - accuracy: 0.6225
Epoch 32/50
25/25 [==============================] - 0s 2ms/step - loss: 0.6477 - accuracy: 0.6237
Epoch 33/50
25/25 [==============================] - 0s 1ms/step - loss: 0.6464 - accuracy: 0.6225
Epoch 34/50
25/25 [==============================] - 0s 1ms/step - loss: 0.6453 - accuracy: 0.6275
```

```
# Evaluate the model on the validation set
loss, accuracy = model.evaluate(X_val, y_val)
print("Validation Accuracy:", accuracy)
```

```
7/7 [==============================] - 0s 2ms/step - loss: 0.7020 - accuracy: 0.5400
Validation Accuracy: 0.5400000214576721
```

Evaluate the accuracy of your model

The accuracy of my model is very similar to the one that I got from the accuracy that I got from the prelim skill exam, since I used the same dataset for this one and I was able to confirm that the result that I got is not a false results. The prediction of the accuracy of the likelihood of injury of the player has a small accuracy or it has a 54% accuracy results for the model so it had a small chance to have an Injury for the player that participated on the data