*# 1. Import the library***import** matplotlib.pyplot **as** plt  
**import** numpy **as** np  
**import** pandas **as** pd  
**from** sklearn.model\_selection **import** train\_test\_split  
**from** tensorflow.keras.layers **import** Dense  
**from** tensorflow.keras.models **import** Sequential  
**from** tensorflow.keras.optimizers **import** RMSprop  
**from** tensorflow.keras.layers **import** Dropout  
  
  
*#2. Load data  
# 2.1 Input data*Input\_data = pd.read\_csv(**"D:/MY LECTURE 2025/DL - Week 2/Input\_data.csv"**)  
*# 2.2 Target Data*Target\_data = pd.read\_csv(**"D:/MY LECTURE 2025/DL - Week 2/Target\_data.csv"**)  
  
*# 3. Pre-processing data  
# 3.1 Convert data to array*Input\_arr = np.array(Input\_data)  
Target\_arr = np.array(Target\_data)  
  
*# 3.2 Transpose the matrix*Input\_tranp = np.transpose(Input\_arr)  
Target\_tranp = np.transpose(Target\_arr)  
  
*# 4. Normalize the data  
# 4.1 formula to normalize***def** norm(x):  
 **return** (x - x.min()) / (x.max() - x.min())  
  
Input\_norm = norm(Input\_tranp)  
  
*# 5 Label the target output  
# 5.1 formula to label the target output***def** label(y):  
 z = (y - y.min())  
 **return** np.round(z)  
Target\_tranp = label(Target\_tranp)  
  
*# 6 Split data 80% for train and 20% for validation*Input\_trai, Input\_valid, Target\_trai, Target\_valid = train\_test\_split(Input\_norm, Target\_tranp, test\_size=0.2)  
  
*# 7. Create deep learning model*ann\_Model = Sequential([  
 Dense(units=250, input\_dim=Input\_trai.shape[1], activation=**'sigmoid'**),  
 Dropout(0.1),  
 Dense(units=500, activation=**'relu'**),  
 Dropout(0.1),  
 Dense(units=500, activation=**'relu'**),  
 Dropout(0.1),  
 Dense(Target\_trai.shape[1], activation=**'elu'**)  
 ])  
  
*# Check model property*print(ann\_Model.summary())  
  
  
*# 8. compile and train model*ann\_Model.compile(loss=**'mean\_squared\_error'**, optimizer=RMSprop(learning\_rate = 0.007), metrics=[**'accuracy'**])  
history = ann\_Model.fit(Input\_trai, Target\_trai, validation\_data=(Input\_valid, Target\_valid), batch\_size=15, epochs=1000, verbose = 1)  
  
*# 9. visualize loss in deep learning*plt.plot(history.history[**'loss'**])  
plt.plot(history.history[**'val\_loss'**])  
plt.title(**'model loss'**)  
plt.ylabel(**'Loss'**)  
plt.xlabel(**'Epoch'**)  
plt.legend([**'train'**, **'val'**], loc=**'upper left'**)  
plt.show()  
  
*# 10. visualize accuracy in deep learning*plt.plot(history.history[**'accuracy'**])  
plt.plot(history.history[**'val\_accuracy'**])  
plt.title(**'model accuracy'**)  
plt.ylabel(**'Accuracy'**)  
plt.xlabel(**'Epoch'**)  
plt.legend([**'train'**, **'val'**], loc=**'upper left'**)  
plt.show()  
  
*# 11. save the models*ann\_Model.save(**'w6\_ANN\_1.h5'**)  
  
*# 12. check the performance of model with actual output and predict output*