Assignment No. 06

Name : Tejas Shyam Fulumbarkar

Subject : Machine Learning

Class : TE-IT (B)

[1]:
```
# !pip install keras tensorflow -U
```

[2]:
```
import math
import numpy as np
import pandas as pd
import tensorflow as tf
import matplotlib.pyplot as plt
from tensorflow.keras import Model
from tensorflow.keras import Sequential
from tensorflow.keras.optimizers import Adam
from sklearn.preprocessing import StandardScaler
from tensorflow.keras.layers import Dense, Dropout
from sklearn.model_selection import train_test_split
from tensorflow.keras.losses import MeanSquaredLogarithmicError
```

[3]:
```
df = np.loadtxt('https://raw.githubusercontent.com/jbrownlee/Datasets/master/pima-indians-diabetes.data.csv', delimiter=',')
```

[4]:
```
df
```

[4]:
```
array([[  6.   , 148.   ,  72.   , ...,   0.627,  50.   ,   1.   ],
       [  1.   ,  85.   ,  66.   , ...,   0.351,  31.   ,   0.   ],
       [  8.   , 183.   ,  64.   , ...,   0.672,  32.   ,   1.   ],
       ...,
       [  5.   , 121.   ,  72.   , ...,   0.245,  30.   ,   0.   ],
       [  1.   , 126.   ,  60.   , ...,   0.349,  47.   ,   1.   ],
       [  1.   ,  93.   ,  70.   , ...,   0.315,  23.   ,   0.   ]])
```

[5]:
```
df.shape
```

[5]:
```
(768, 9)
```

[6]:
```
x = df[:,:8]
y = df[:,8]
```

[7]:
```
from sklearn.model_selection import train_test_split
X_train, X_temp, y_train, y_temp = train_test_split(x, y, test_size=0.2, random_state=42)

X_test, X_val, y_test, y_val = train_test_split(X_temp, y_temp, test_size=0.5, random_state=42)
```

[8]:
```
print(f"x train shape{X_train.shape}")
print(f"y train shape{y_train.shape}")
print(f"x test shape{X_test.shape}")
print(f"y test shape{y_test.shape}")
print(f"x val shape{X_val.shape}")
print(f"y val shape{y_val.shape}")
```

```
x train shape(614, 8)
y train shape(614,)
x test shape(77, 8)
y test shape(77,)
x val shape(77, 8)
y val shape(77,)
```

[9]:
```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
X_val = scaler.transform(X_val)
```

[10]:
```
from collections import Counter
Counter(y)
```

10]:
```
Counter({1.0: 268, 0.0: 500})
```

11]:
```
import seaborn as sns
```

[12]:
```
# sns.countplot(y)
```

[13]:
```
from tensorflow.keras.models import Sequential
```

[14]:
```
model = Sequential([

tf.keras.layers.InputLayer(8,),
Dense(50,activation='relu'),

Dense(50,activation='relu'),
Dense(50,activation='relu'),
Dense(50,activation='relu'),

Dense(1,activation='sigmoid')
])
```

[15]:
```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 50)                450
_________________________________________________________________
dense_1 (Dense)              (None, 50)                2550
_________________________________________________________________
dense_2 (Dense)              (None, 50)                2550
_________________________________________________________________
dense_3 (Dense)              (None, 50)                2550
_________________________________________________________________
dense_4 (Dense)              (None, 1)                 51
=================================================================
Total params: 8,151
Trainable params: 8,151
Non-trainable params: 0
_________________________________________________________________
```

[16]:
```
opt = tf.keras.optimizers.Adam(learning_rate=0.0001)
model.compile(loss='binary_crossentropy', optimizer=opt, metrics=['accuracy'])
```

[17]:
```
history = model.fit(x=x,y=y,epochs=300, batch_size=50,validation_data=(X_val,y_val))
```

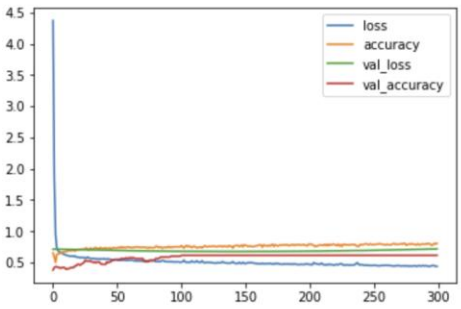
```
2022-11-12 19:23:28.666100: I tensorflow/compiler/mlir/mlir_graph_optimization_pass.cc:185] None of the MLIR Optimization Pas
ses are enabled (registered 2)
Epoch 1/300
16/16 [==============================] - 1s 20ms/step - loss: 4.3687 - accuracy: 0.6484 - val_loss: 0.7085 - val_accuracy: 0.
3766
Epoch 2/300
16/16 [==============================] - 0s 5ms/step - loss: 1.8559 - accuracy: 0.5820 - val_loss: 0.7094 - val_accuracy: 0.4
156
Epoch 3/300
16/16 [==============================] - 0s 5ms/step - loss: 0.9362 - accuracy: 0.4987 - val_loss: 0.7087 - val_accuracy: 0.4
286
Epoch 4/300
16/16 [==============================] - 0s 5ms/step - loss: 0.7237 - accuracy: 0.6276 - val_loss: 0.7089 - val_accuracy: 0.4
286
Epoch 5/300
16/16 [==============================] - 0s 6ms/step - loss: 0.6846 - accuracy: 0.6341 - val_loss: 0.7092 - val_accuracy: 0.4
156
Epoch 6/300
16/16 [==============================] - 0s 5ms/step - loss: 0.6645 - accuracy: 0.6406 - val_loss: 0.7088 - val_accuracy: 0.4
156
Epoch 7/300
16/16 [==============================] - 0s 5ms/step - loss: 0.6514 - accuracy: 0.6471 - val_loss: 0.7083 - val_accuracy: 0.4
026
Epoch 8/300
16/16 [==============================] - 0s 5ms/step - loss: 0.6433 - accuracy: 0.6562 - val_loss: 0.7080 - val_accuracy: 0.4
156
Epoch 9/300
16/16 [==============================] - 0s 5ms/step - loss: 0.6350 - accuracy: 0.6641 - val_loss: 0.7077 - val_accuracy: 0.4
156

Epoch 10/300
16/16 [==============================] - 0s 5ms/step - loss: 0.6243 - accuracy: 0.6706 - val_loss: 0.7072 - val_accuracy: 0.4
156
Epoch 11/300
16/16 [==============================] - 0s 5ms/step - loss: 0.6146 - accuracy: 0.6667 - val_loss: 0.7070 - val_accuracy: 0.3
896
Epoch 12/300
16/16 [==============================] - 0s 5ms/step - loss: 0.6075 - accuracy: 0.6849 - val_loss: 0.7066 - val_accuracy: 0.3
896
Epoch 13/300
16/16 [==============================] - 0s 5ms/step - loss: 0.6069 - accuracy: 0.6823 - val_loss: 0.7061 - val_accuracy: 0.4
026
Epoch 14/300
16/16 [==============================] - 0s 5ms/step - loss: 0.5973 - accuracy: 0.6888 - val_loss: 0.7054 - val_accuracy: 0.4
026
Epoch 15/300
16/16 [==============================] - 0s 5ms/step - loss: 0.5979 - accuracy: 0.6823 - val_loss: 0.7049 - val_accuracy: 0.4
156
Epoch 16/300
16/16 [==============================] - 0s 5ms/step - loss: 0.6024 - accuracy: 0.6901 - val_loss: 0.7043 - val_accuracy: 0.4
156
Epoch 17/300
16/16 [==============================] - 0s 6ms/step - loss: 0.5915 - accuracy: 0.6927 - val_loss: 0.7039 - val_accuracy: 0.4
156
Epoch 18/300
16/16 [==============================] - 0s 6ms/step - loss: 0.6017 - accuracy: 0.6797 - val_loss: 0.7033 - val_accuracy: 0.4
416
Epoch 19/300
16/16 [==============================] - 0s 5ms/step - loss: 0.5847 - accuracy: 0.6888 - val_loss: 0.7029 - val_accuracy: 0.4
416
Epoch 20/300
16/16 [==============================] - 0s 5ms/step - loss: 0.5832 - accuracy: 0.7044 - val_loss: 0.7021 - val_accuracy: 0.4
675
```

```
Epoch 34/300
16/16 [==============================] - 0s 6ms/step - loss: 0.5540 - accuracy: 0.7357 - val_loss: 0.6963 - val_accuracy: 0.5
065
Epoch 35/300
16/16 [==============================] - 0s 5ms/step - loss: 0.5552 - accuracy: 0.7214 - val_loss: 0.6960 - val_accuracy: 0.4
935
Epoch 36/300
16/16 [==============================] - 0s 5ms/step - loss: 0.5528 - accuracy: 0.7279 - val_loss: 0.6955 - val_accuracy: 0.5
065
Epoch 37/300
16/16 [==============================] - 0s 5ms/step - loss: 0.5540 - accuracy: 0.7174 - val_loss: 0.6952 - val_accuracy: 0.5
065
Epoch 38/300
16/16 [==============================] - 0s 5ms/step - loss: 0.5522 - accuracy: 0.7357 - val_loss: 0.6950 - val_accuracy: 0.4
805
Epoch 39/300
16/16 [==============================] - 0s 5ms/step - loss: 0.5536 - accuracy: 0.7214 - val_loss: 0.6945 - val_accuracy: 0.4
675
Epoch 40/300
16/16 [==============================] - 0s 6ms/step - loss: 0.5521 - accuracy: 0.7266 - val_loss: 0.6941 - val_accuracy: 0.4
675
Epoch 41/300
16/16 [==============================] - 0s 7ms/step - loss: 0.5580 - accuracy: 0.7201 - val_loss: 0.6935 - val_accuracy: 0.4
675
Epoch 42/300
16/16 [==============================] - 0s 5ms/step - loss: 0.5480 - accuracy: 0.7331 - val_loss: 0.6930 - val_accuracy: 0.4
675
Epoch 43/300
16/16 [==============================] - 0s 5ms/step - loss: 0.5494 - accuracy: 0.7214 - val_loss: 0.6927 - val_accuracy: 0.5
065
Epoch 47/300
16/16 [==============================] - 0s 6ms/step - loss: 0.5392 - accuracy: 0.7292 - val_loss: 0.6909 - val_accuracy: 0.5
325
Epoch 48/300
16/16 [==============================] - 0s 5ms/step - loss: 0.5374 - accuracy: 0.7344 - val_loss: 0.6908 - val_accuracy: 0.5
325
Epoch 49/300
16/16 [==============================] - 0s 6ms/step - loss: 0.5382 - accuracy: 0.7266 - val_loss: 0.6904 - val_accuracy: 0.5
455
Epoch 50/300
16/16 [==============================] - 0s 5ms/step - loss: 0.5440 - accuracy: 0.7279 - val_loss: 0.6902 - val_accuracy: 0.5
455
Epoch 51/300
16/16 [==============================] - 0s 6ms/step - loss: 0.5350 - accuracy: 0.7305 - val_loss: 0.6898 - val_accuracy: 0.5
455
Epoch 52/300
16/16 [==============================] - 0s 5ms/step - loss: 0.5325 - accuracy: 0.7344 - val_loss: 0.6893 - val_accuracy: 0.5
455
Epoch 53/300
16/16 [==============================] - 0s 6ms/step - loss: 0.5316 - accuracy: 0.7318 - val_loss: 0.6887 - val_accuracy: 0.5
455
Epoch 54/300
16/16 [==============================] - 0s 5ms/step - loss: 0.5328 - accuracy: 0.7487 - val_loss: 0.6885 - val_accuracy: 0.5
584
Epoch 55/300
16/16 [==============================] - 0s 5ms/step - loss: 0.5327 - accuracy: 0.7305 - val_loss: 0.6882 - val_accuracy: 0.5
584
Epoch 56/300
16/16 [==============================] - 0s 5ms/step - loss: 0.5335 - accuracy: 0.7422 - val_loss: 0.6879 - val_accuracy: 0.5
584
```

[18]:
```
losses = pd.DataFrame(model.history.history)
losses.plot()
```

[18]: <AxesSubplot:>



[19]:
```
model.evaluate(x,y)
```

```
24/24 [==============================] - 0s 2ms/step - loss: 0.4277 - accuracy: 0.8099
```

[19]: [0.42774689197540283, 0.8098958134651184]

[20]:
```
y_pred = model.predict(X_test)
```

[20]:
```
y_pred = model.predict(X_test)
```

[21]:
```
y_pred
```

[21]:
```
array([[0.29241186],
       [0.25659525],
       [0.2345556 ],
       [0.25635856],
       [0.2683779 ],
       [0.26534295],
       [0.24615443],
       [0.18696874],
       [0.23531413],
       [0.26499653],
       [0.25147843],
       [0.19693151],
       [0.2525043 ],
       [0.2924193 ],
       [0.24108097],
       [0.24926525],
       [0.24154398],
       [0.30027997],
       [0.26890767],
       [0.18842933],
       [0.27719277],
       [0.23296693],
       [0.25262323],
       [0.23384169],
       [0.26180875],
       [0.2531795 ],
```

```
[0.23095623],
[0.24577609],
[0.21299466],
[0.25943238],
[0.26748422],
[0.28173897],
[0.21895227],
[0.25581098],
[0.2772973 ],
[0.26322144],
[0.27639455],
[0.2651384 ],
[0.29406345],
[0.19382304],
[0.24121556],
[0.2256059 ],
[0.2787133 ],
[0.28141072],
[0.22622645],
[0.30917984],
[0.23824352],
[0.25481486],
[0.25906867],
[0.2787506 ],
[0.20846003],
[0.2344867 ],
[0.27582753],
[0.24831259],
[0.29489657],
[0.26952785],
[0.26609173],
[0.26449886],
[0.3253206 ],
```

[22]:
```
# !pip install ann_visualizer
```

[23]:
```
# !pip install graphviz
```

[24]:
```
# from ann_visualizer.visualize import ann_viz

# ann_viz(model, title="")
```

[25]:
```
# !pip3 install keras
# !pip3 install ann_visualizer
# !pip install graphviz
```

[26]:
```
# from ann_visualizer.visualize import ann_viz;

# ann_viz(model, title="My first neural network")
```