Assignment No. 1

Aim: Introduction to Keras and Tensorflow (Optional – Pytorch). Configure and use google colab and kaggle GPU

Objectives:

1. To configure anaconda and google colab, kaggle environment
2. To Explore TF/Keras/Pytorch libraries
3. To learn to use GPU/TPU
4. To learn and understand Git

Theory:

Keras Configuration

1. Setup Environment
2. pip install keras
3. to check whether it has installed properly: python>>import keras

Python>> print keras.__version__

Tensorflow Configuration:

1. pip install tensorflow==2.2.0
2. To verify installation: python>> import tensorflow as tf

If no error then the installation has been completed

Colaboratory Configuration

1. Setup the environment
2. Connect to the Drive
3. Upload the files using: from google.colab import files

files.upload()

Kaggle Configuration:

1. Create a Kaggle Account
2. Create an Authorization token
3. Upload on Colab using the upload code
4. Make a folder and make the Json file executable
5. Get the dataset

6. Copy the API command and run on colab

GitHub Configuration

1. pip install git
2. Set up user profile by: global user.name “name”

   Global user.email “mail”

Dataset Attributes:

1. Pregnancies
2. Glucose
3. BloodPressure
4. SkinThickness
5. Insulin
6. BMI
7. DiabetesPedigreeFunction
8. Age
9. Outcome

Code:

```
import pandas as pd
import numpy as np
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers,models
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler


df=pd.read_csv("diabetes.csv")
x=df.iloc[:,0:8]
y=df["Outcome"]
obj=StandardScaler()
x_=obj.fit_transform(x)
Xtrain,Xtest,Ytrain,Ytest=train_test_split(x_,y,test_size=0.1)


model=models.Sequential()
model.add(layers.Dense(100,activation="relu"))
```

```
#model.add(layers.Dense(75,activation="relu"))
model.add(layers.Dense(50,activation="relu"))
#model.add(layers.Dense(25,activation="relu"))
model.add(layers.Dense(12,activation="relu"))
model.add(layers.Dense(8,activation="relu"))
model.add(layers.Dense(1,activation="sigmoid"))


model.compile(optimizer="adam",loss="binary_crossentropy",metrics=["accuracy"]
,)
history=model.fit(Xtrain,Ytrain,epochs=50, validation_data=(Xtest,Ytest))
result=model.evaluate(Xtest,Ytest)


import matplotlib.pyplot as plt
plt.plot(history.history['loss'], label='loss')
#plt.plot(history.history['val_accuracy'], label = 'val_accuracy')
plt.xlabel('Epoch')
plt.ylabel('loss')
plt.ylim([0, 0.8])
plt.legend(loc='lower right')

test_loss, test_acc = model.evaluate(Xtest, Ytest, verbose=2)
plt.ylim([0.6,1])
plt.plot(history.history['accuracy'], label = 'accuracy')
```

Results:

Training:

```
Epoch 1/50
22/22 [==============================] - 1s 23ms/step - loss: 0.6561 - accuracy: 0.6483 - val_loss: 0.5546 - val_accuracy: 0.6753
Epoch 2/50
22/22 [==============================] - 0s 6ms/step - loss: 0.5653 - accuracy: 0.6483 - val_loss: 0.4855 - val_accuracy: 0.6753
Epoch 3/50
22/22 [==============================] - 0s 6ms/step - loss: 0.5234 - accuracy: 0.6483 - val_loss: 0.4770 - val_accuracy: 0.6753
Epoch 4/50
22/22 [==============================] - 0s 7ms/step - loss: 0.5087 - accuracy: 0.6527 - val_loss: 0.4726 - val_accuracy: 0.7403
Epoch 5/50
22/22 [==============================] - 0s 7ms/step - loss: 0.4994 - accuracy: 0.7496 - val_loss: 0.4736 - val_accuracy: 0.8052
Epoch 6/50
22/22 [==============================] - 0s 9ms/step - loss: 0.4920 - accuracy: 0.7974 - val_loss: 0.4664 - val_accuracy: 0.8182
Epoch 7/50
22/22 [==============================] - 0s 7ms/step - loss: 0.4873 - accuracy: 0.7902 - val_loss: 0.4674 - val_accuracy: 0.8312
Epoch 8/50
22/22 [==============================] - 0s 7ms/step - loss: 0.4815 - accuracy: 0.7916 - val_loss: 0.4685 - val_accuracy: 0.8052
Epoch 9/50
22/22 [==============================] - 0s 7ms/step - loss: 0.4711 - accuracy: 0.7916 - val_loss: 0.4599 - val_accuracy: 0.8312
Epoch 10/50
22/22 [==============================] - 0s 7ms/step - loss: 0.4546 - accuracy: 0.7931 - val_loss: 0.4447 - val_accuracy: 0.8052
Epoch 11/50
22/22 [==============================] - 0s 7ms/step - loss: 0.4422 - accuracy: 0.7873 - val_loss: 0.4392 - val_accuracy: 0.7922
Epoch 12/50
22/22 [==============================] - 0s 7ms/step - loss: 0.4281 - accuracy: 0.8032 - val_loss: 0.4383 - val_accuracy: 0.8182
Epoch 13/50
```

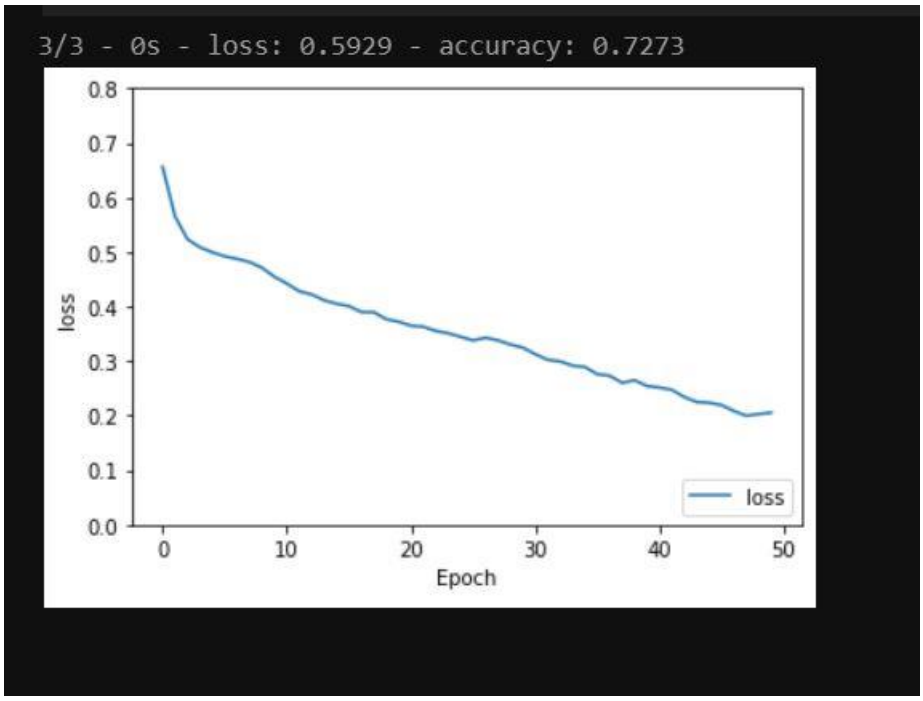
```
22/22 [==============================] - 0s 7ms/step - loss: 0.2601 - accuracy: 0.9030 - val_loss: 0.4941 - val_accuracy: 0.7792
Epoch 39/50
22/22 [==============================] - 0s 7ms/step - loss: 0.2649 - accuracy: 0.9045 - val_loss: 0.4924 - val_accuracy: 0.7532
Epoch 40/50
22/22 [==============================] - 0s 7ms/step - loss: 0.2543 - accuracy: 0.9074 - val_loss: 0.5258 - val_accuracy: 0.7532
Epoch 41/50
22/22 [==============================] - 0s 8ms/step - loss: 0.2515 - accuracy: 0.8958 - val_loss: 0.5441 - val_accuracy: 0.7143
Epoch 42/50
22/22 [==============================] - 0s 7ms/step - loss: 0.2473 - accuracy: 0.9103 - val_loss: 0.5321 - val_accuracy: 0.7403
Epoch 43/50
22/22 [==============================] - 0s 10ms/step - loss: 0.2341 - accuracy: 0.9204 - val_loss: 0.5547 - val_accuracy: 0.7273
Epoch 44/50
22/22 [==============================] - 0s 9ms/step - loss: 0.2249 - accuracy: 0.9233 - val_loss: 0.5368 - val_accuracy: 0.7792
Epoch 45/50
22/22 [==============================] - 0s 10ms/step - loss: 0.2236 - accuracy: 0.9190 - val_loss: 0.5579 - val_accuracy: 0.7403
Epoch 46/50
22/22 [==============================] - 0s 8ms/step - loss: 0.2193 - accuracy: 0.9204 - val_loss: 0.5855 - val_accuracy: 0.7143
Epoch 47/50
22/22 [==============================] - 0s 9ms/step - loss: 0.2084 - accuracy: 0.9305 - val_loss: 0.5877 - val_accuracy: 0.7273
Epoch 48/50
22/22 [==============================] - 0s 8ms/step - loss: 0.1997 - accuracy: 0.9334 - val_loss: 0.5976 - val_accuracy: 0.7013
Epoch 49/50
22/22 [==============================] - 0s 6ms/step - loss: 0.2029 - accuracy: 0.9363 - val_loss: 0.5827 - val_accuracy: 0.7143
Epoch 50/50
22/22 [==============================] - 0s 7ms/step - loss: 0.2054 - accuracy: 0.9334 - val_loss: 0.5929 - val_accuracy: 0.7273
```

Evaluation

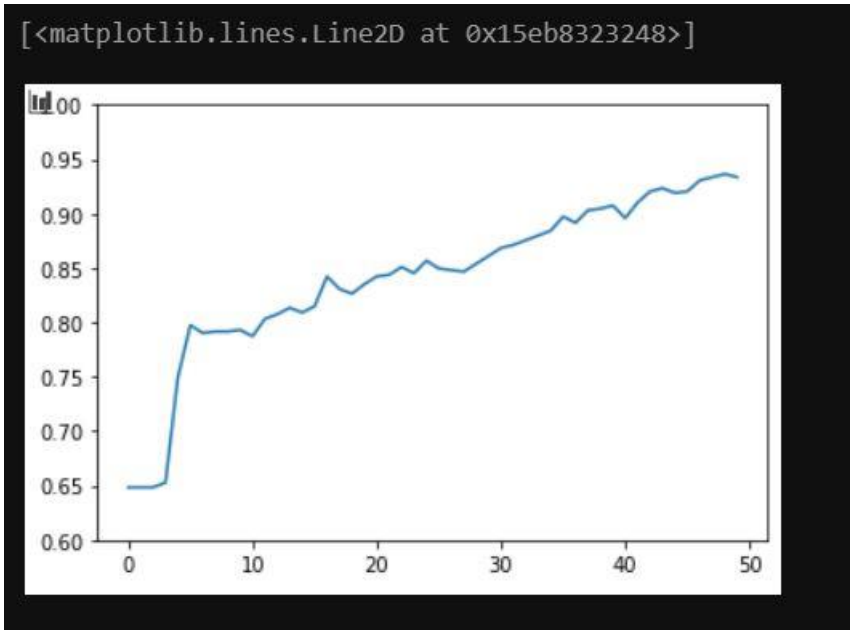
```
result=model.evaluate(Xtest,Ytest)
```

```
3/3 [==============================] - 0s 3ms/step - loss: 0.5929 - accuracy: 0.7273
```

Loss function



Accuracy



Conclusion:

Thus we have understood the configuration steps of Google Colab, tensorflow, etc and learned to use tensorflow and create a model to predict the chance of diabetes or not.