This dataset is originally from the National Institute of Diabetes and Digestive and Kidney Diseases. Predict if a person has diabetes or not, using single-neuron classification.

Acknowledgment - kaggle.com

Data - drive.google.com

Features -

Pregnancies
Number of times pregnant

Glucose
Plasma glucose concentration a 2 hours in an oral glucose tolerance test

BloodPressure
Diastolic blood pressure (mm Hg)

SkinThickness
Triceps skin fold thickness (mm)

Insulin
2-Hour serum insulin (mu U/ml)

BMI
Body mass index (weight in kg/(height in m)^2)

DiabetesPedigreeFunction
Diabetes pedigree function

```
Age
Age (years)




Outcome
Class variable (0 or 1) 268 of 768 are 1, the others are 0
```

In [6]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')
import tensorflow as tf
from sklearn.model_selection import train_test_split
from sklearn.metrics import roc_auc_score
from sklearn.preprocessing import StandardScaler
```

In [7]:

```
df=pd.read_csv("diabetes.csv")
```

In [8]:

```
df
```

Out[8]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunct
0	6	148	72	35	0	33.6	0.
1	1	85	66	29	0	26.6	0.
2	8	183	64	0	0	23.3	0.
3	1	89	66	23	94	28.1	0.
4	0	137	40	35	168	43.1	2.
...	...	...	...	...	...	...	
763	10	101	76	48	180	32.9	0.
764	2	122	70	27	0	36.8	0.
765	5	121	72	23	112	26.2	0.
766	1	126	60	0	0	30.1	0.
767	1	93	70	31	0	30.4	0.

768 rows × 9 columns

In [9]:

```
x=df.drop("Outcome",axis=1)
y=df["Outcome"]
```

In [10]:

```
ss=StandardScaler()
x = ss.fit_transform(x)
```

In [13]:

```
x.shape[1]
```

Out[13]:

```
8
```

In [12]:

```
x_train,x_test, y_train, y_test = train_test_split(x, y, test_size=0.3)
```

In [15]:

```
model = tf.keras.Sequential([tf.keras.layers.Dense(2, activation="relu",input_shape=(x.shap
                             tf.keras.layers.Dense(3, activation="relu"),
                             tf.keras.layers.Dense(1, activation="sigmoid")]
                            )
```

In [17]:

```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
dense (Dense)                (None, 2)                 18        
_________________________________________________________________
dense_1 (Dense)              (None, 3)                 9         
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 4         
=================================================================
Total params: 31
Trainable params: 31
Non-trainable params: 0
_________________________________________________________________
```

In [18]:

```
model.compile(optimizer="sgd", loss="binary_crossentropy")
```

In [19]:

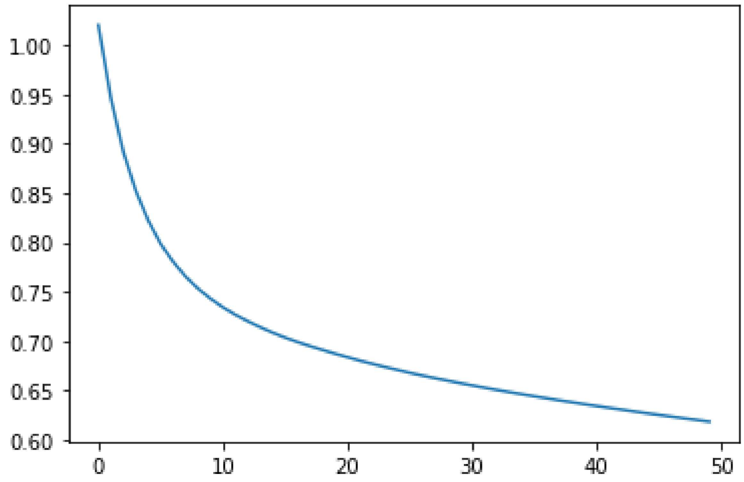
```
trained_model = model.fit(x_train, y_train, epochs=50, batch_size=50)
```

```
Epoch 1/50
11/11 [==============================] - 1s 799us/step - loss: 1.0585
Epoch 2/50
11/11 [==============================] - 0s 2ms/step - loss: 0.9285
Epoch 3/50
11/11 [==============================] - 0s 2ms/step - loss: 0.8698
Epoch 4/50
11/11 [==============================] - 0s 2ms/step - loss: 0.8494
Epoch 5/50
11/11 [==============================] - 0s 800us/step - loss: 0.8068
Epoch 6/50
11/11 [==============================] - 0s 1ms/step - loss: 0.7929
Epoch 7/50
11/11 [==============================] - 0s 1ms/step - loss: 0.7669
Epoch 8/50
11/11 [==============================] - 0s 2ms/step - loss: 0.7674
Epoch 9/50
11/11 [==============================] - 0s 799us/step - loss: 0.7446
Epoch 10/50
11/11 [==============================] - 0s 1ms/step - loss: 0.7700
Epoch 11/50
11/11 [==============================] - 0s 1ms/step - loss: 0.7347
Epoch 12/50
11/11 [==============================] - 0s 1ms/step - loss: 0.7443
Epoch 13/50
11/11 [==============================] - 0s 1ms/step - loss: 0.7337
Epoch 14/50
11/11 [==============================] - 0s 1ms/step - loss: 0.7383
Epoch 15/50
11/11 [==============================] - 0s 2ms/step - loss: 0.7100
Epoch 16/50
11/11 [==============================] - 0s 1ms/step - loss: 0.6954
Epoch 17/50
11/11 [==============================] - 0s 1ms/step - loss: 0.6971
Epoch 18/50
11/11 [==============================] - 0s 2ms/step - loss: 0.7126
Epoch 19/50
11/11 [==============================] - 0s 1ms/step - loss: 0.6935
Epoch 20/50
11/11 [==============================] - 0s 1ms/step - loss: 0.6936
Epoch 21/50
11/11 [==============================] - 0s 2ms/step - loss: 0.6824
Epoch 22/50
11/11 [==============================] - 0s 2ms/step - loss: 0.6782
Epoch 23/50
11/11 [==============================] - 0s 2ms/step - loss: 0.6590
Epoch 24/50
11/11 [==============================] - 0s 2ms/step - loss: 0.6659
Epoch 25/50
11/11 [==============================] - 0s 2ms/step - loss: 0.6578
Epoch 26/50
11/11 [==============================] - 0s 2ms/step - loss: 0.6646
Epoch 27/50
11/11 [==============================] - 0s 2ms/step - loss: 0.6617
Epoch 28/50
11/11 [==============================] - 0s 2ms/step - loss: 0.6728
```

```
Epoch 29/50
11/11 [==============================] - 0s 2ms/step - loss: 0.6448
Epoch 30/50
11/11 [==============================] - 0s 2ms/step - loss: 0.6475
Epoch 31/50
11/11 [==============================] - 0s 2ms/step - loss: 0.6400
Epoch 32/50
11/11 [==============================] - 0s 2ms/step - loss: 0.6357
Epoch 33/50
11/11 [==============================] - 0s 2ms/step - loss: 0.6489
Epoch 34/50
11/11 [==============================] - 0s 2ms/step - loss: 0.6563
Epoch 35/50
11/11 [==============================] - 0s 1ms/step - loss: 0.6516
Epoch 36/50
11/11 [==============================] - 0s 2ms/step - loss: 0.6411
Epoch 37/50
11/11 [==============================] - 0s 1ms/step - loss: 0.6395
Epoch 38/50
11/11 [==============================] - 0s 2ms/step - loss: 0.6343
Epoch 39/50
11/11 [==============================] - 0s 2ms/step - loss: 0.6343
Epoch 40/50
11/11 [==============================] - 0s 2ms/step - loss: 0.6216
Epoch 41/50
11/11 [==============================] - 0s 2ms/step - loss: 0.6345
Epoch 42/50
11/11 [==============================] - 0s 2ms/step - loss: 0.6293
Epoch 43/50
11/11 [==============================] - 0s 2ms/step - loss: 0.6471
Epoch 44/50
11/11 [==============================] - 0s 1ms/step - loss: 0.6287
Epoch 45/50
11/11 [==============================] - 0s 2ms/step - loss: 0.6230
Epoch 46/50
11/11 [==============================] - 0s 1ms/step - loss: 0.6383
Epoch 47/50
11/11 [==============================] - 0s 1ms/step - loss: 0.6221
Epoch 48/50
11/11 [==============================] - 0s 1ms/step - loss: 0.6129
Epoch 49/50
11/11 [==============================] - 0s 2ms/step - loss: 0.6136
Epoch 50/50
11/11 [==============================] - 0s 1ms/step - loss: 0.6131
```

In [20]:

```
plt.plot(trained_model.history['loss'])
plt.show()
```



In [21]:

```
y_hat = model.predict(x_test)
```

In [22]:

```
y_hat
```

Out[22]:

```
array([[0.4241799 ],
       [0.45988527],
       [0.2993986 ],
       [0.42947233],
       [0.45988527],
       [0.45988527],
       [0.45645726],
       [0.24880001],
       [0.25102407],
       [0.45988527],
       [0.31837046],
       [0.4568506 ],
       [0.45988527],
       [0.31209302],
       [0.42734456],
       [0.30262434],
       [0.45988527],
       [0.44010437],
```

In [23]:

```
y_hat1 = np.where(y_hat >= 0.5, 1, 0) #where value is o.5 than greater then shows 1 else 0
```

In [24]:

```
y_hat1.flatten()
```

Out[24]:

```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0])
```

In [25]:

```
roc_auc_score(y_test, y_hat1)
```

Out[25]:

```
0.5
```

In [ ]:

```

```