ANN_Binary_Diabeties

```
from numpy import loadtxt
import tensorflow as tf
import pandas as pd
from keras.models import Sequential
from keras.layers import Dense
```

```
dataset = pd.read_csv('/content/diabetes2.csv')
```

```
dataset.isnull().any()
dataset = dataset.fillna(method='ffill')
```

```
X = dataset[['Pregnancies','Glucose','BloodPressure','SkinThickness','Insulin','BMI','DiabetesPedigreeFunction','Age']].values
y = dataset['Outcome'].values
```

```
model = Sequential([
        Dense(12, input_shape=(8,),activation='relu'),
        Dense(8, activation='relu'),
        Dense(1,activation='sigmoid')])
```

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

```
history = model.fit(X,y,validation_split=0.33, epochs =300, batch_size=5, verbose= 1)
```

```
Epoch 1/300
103/103 [==============================] - 1s 6ms/step - loss: 3.2927 - accuracy: 0.4650 - val_loss: 1.5679 - val_accuracy: 0.476
Epoch 2/300
103/103 [==============================] - 0s 4ms/step - loss: 1.4333 - accuracy: 0.5700 - val_loss: 1.0660 - val_accuracy: 0.618
Epoch 3/300
103/103 [==============================] - 0s 5ms/step - loss: 1.0426 - accuracy: 0.5953 - val_loss: 0.8291 - val_accuracy: 0.673
Epoch 4/300
103/103 [==============================] - 0s 5ms/step - loss: 0.9710 - accuracy: 0.6167 - val_loss: 1.0742 - val_accuracy: 0.618
Epoch 5/300
103/103 [==============================] - 0s 4ms/step - loss: 0.8846 - accuracy: 0.6284 - val_loss: 0.7509 - val_accuracy: 0.566
Epoch 6/300
103/103 [==============================] - 0s 5ms/step - loss: 0.8827 - accuracy: 0.6226 - val_loss: 0.7372 - val_accuracy: 0.673
Epoch 7/300
103/103 [==============================] - 0s 4ms/step - loss: 0.7597 - accuracy: 0.6712 - val_loss: 0.6667 - val_accuracy: 0.661
Epoch 8/300
103/103 [==============================] - 0s 4ms/step - loss: 0.7668 - accuracy: 0.6595 - val_loss: 0.7299 - val_accuracy: 0.618
Epoch 9/300
103/103 [==============================] - 0s 5ms/step - loss: 0.6851 - accuracy: 0.6732 - val_loss: 0.7036 - val_accuracy: 0.574
Epoch 10/300
103/103 [==============================] - 0s 5ms/step - loss: 0.7076 - accuracy: 0.6848 - val_loss: 0.6597 - val_accuracy: 0.637
Epoch 11/300
103/103 [==============================] - 0s 4ms/step - loss: 0.6692 - accuracy: 0.6984 - val_loss: 0.7209 - val_accuracy: 0.696
Epoch 12/300
103/103 [==============================] - 0s 5ms/step - loss: 0.7291 - accuracy: 0.6615 - val_loss: 0.6558 - val_accuracy: 0.637
Epoch 13/300
103/103 [==============================] - 0s 4ms/step - loss: 0.7041 - accuracy: 0.6576 - val_loss: 0.7056 - val_accuracy: 0.622
Epoch 14/300
103/103 [==============================] - 0s 5ms/step - loss: 0.7374 - accuracy: 0.6498 - val_loss: 0.6219 - val_accuracy: 0.720
Epoch 15/300
103/103 [==============================] - 0s 5ms/step - loss: 0.6628 - accuracy: 0.6887 - val_loss: 0.6579 - val_accuracy: 0.633
Epoch 16/300
103/103 [==============================] - 0s 5ms/step - loss: 0.6554 - accuracy: 0.7140 - val_loss: 0.6022 - val_accuracy: 0.712
Epoch 17/300
103/103 [==============================] - 0s 4ms/step - loss: 0.6694 - accuracy: 0.6732 - val_loss: 0.6665 - val_accuracy: 0.696
Epoch 18/300
103/103 [==============================] - 0s 5ms/step - loss: 0.7099 - accuracy: 0.6615 - val_loss: 0.6332 - val_accuracy: 0.712
Epoch 19/300
103/103 [==============================] - 1s 6ms/step - loss: 0.7219 - accuracy: 0.6654 - val_loss: 0.8562 - val_accuracy: 0.563
Epoch 20/300
103/103 [==============================] - 1s 6ms/step - loss: 0.7787 - accuracy: 0.6615 - val_loss: 0.7454 - val_accuracy: 0.673
Epoch 21/300
103/103 [==============================] - 1s 6ms/step - loss: 0.6601 - accuracy: 0.6907 - val_loss: 0.6935 - val_accuracy: 0.618
Epoch 22/300
103/103 [==============================] - 1s 6ms/step - loss: 0.6835 - accuracy: 0.6984 - val_loss: 0.6343 - val_accuracy: 0.661
Epoch 23/300
103/103 [==============================] - 1s 5ms/step - loss: 0.6443 - accuracy: 0.7179 - val_loss: 0.6480 - val_accuracy: 0.665
Epoch 24/300
103/103 [==============================] - 0s 5ms/step - loss: 0.6209 - accuracy: 0.6887 - val_loss: 0.6359 - val_accuracy: 0.689
Epoch 25/300
103/103 [==============================] - 0s 5ms/step - loss: 0.6602 - accuracy: 0.6829 - val_loss: 0.6228 - val_accuracy: 0.700
Epoch 26/300
103/103 [==============================] - 0s 5ms/step - loss: 0.6233 - accuracy: 0.7082 - val_loss: 0.6269 - val_accuracy: 0.657
Epoch 27/300
103/103 [==============================] - 0s 5ms/step - loss: 0.6395 - accuracy: 0.6887 - val_loss: 0.8517 - val_accuracy: 0.696
```

```
Epoch 28/300
103/103 [==============================] - 0s 4ms/step - loss: 0.6890 - accuracy: 0.6965 - val_loss: 0.6170 - val_accuracy: 0.700
```

```
_, accuracy = model.evaluate(X,y)
print('Accuracy: %.2f'%(accuracy*100))
```

```
24/24 [==============================] - 0s 3ms/step - loss: 0.6108 - accuracy: 0.7018
Accuracy: 70.18
```

```
import matplotlib.pyplot as plt
print(history.history.keys())
#sumarise history from accuracy
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train','test'],loc='upper left')
plt.show()
#sumarise history for loss
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train','test'],loc='upper left')
plt.show()
```

```
dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy'])
```