Artificial Neural Network

Importing the libraries

```
import numpy as np
import pandas as pd
import tensorflow as tf
```

```
tf.__version__
```

```
'2.7.0'
```

Part 1 - Data Preprocessing

Importing the dataset

```
data = pd.read_csv('Cancer.csv')
data.head()
#X = dataset.iloc[:, 3:-1].values
#y = dataset.iloc[:, -1].values
```

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothnes
0	842302	M	17.99	10.38	122.80	1001.0	(
1	842517	M	20.57	17.77	132.90	1326.0	(
2	84300903	M	19.69	21.25	130.00	1203.0	(
3	84348301	M	11.42	20.38	77.58	386.1	(
4	84358402	M	20.29	14.34	135.10	1297.0	(

Encoding categorical data

Label Encoding the "Gender" column

```
from sklearn.preprocessing import LabelEncoder
le_diagnosis = LabelEncoder()
data['diagnosis_n'] = le_diagnosis.fit_transform(data['diagnosis'])
data.head()
```

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothnes
0	842302	M	17.99	10.38	122.80	1001.0	(
1	842517	M	20.57	17.77	132.90	1326.0	(
2	84300903	M	19.69	21.25	130.00	1203.0	(
3	84348301	M	11.42	20.38	77.58	386.1	(
4	84358402	M	20.29	14.34	135.10	1297.0	(

```
data = data.drop(['diagnosis'],axis='columns')
data.head()
```

	id	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	com
0	842302	17.99	10.38	122.80	1001.0	0.11840	
1	842517	20.57	17.77	132.90	1326.0	0.08474	
2	84300903	19.69	21.25	130.00	1203.0	0.10960	
3	84348301	11.42	20.38	77.58	386.1	0.14250	
4	84358402	20.29	14.34	135.10	1297.0	0.10030	

```
#Define What X and y are in our Dataset
y = data['diagnosis_n']
X = data.drop('diagnosis_n', axis = 1)
print(X.head())
print(y.head())
```

```
         id  radius_mean  ...  fractal_dimension_worst  Unnamed: 32
0    842302        17.99  ...                  0.11890          NaN
1    842517        20.57  ...                  0.08902          NaN
2  84300903        19.69  ...                  0.08758          NaN
3  84348301        11.42  ...                  0.17300          NaN
4  84358402        20.29  ...                  0.07678          NaN

[5 rows x 32 columns]
0    1
1    1
2    1
```

```
3    1
4    1
Name: diagnosis_n, dtype: int64
```

Feature Scaling

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X = sc.fit_transform(X)
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/utils/extmath.py:986: RuntimeWarning: inv
  updated_mean = (last_sum + new_sum) / updated_sample_count
/usr/local/lib/python3.7/dist-packages/sklearn/utils/extmath.py:991: RuntimeWarning: inv
  T = new_sum / new_sample_count
/usr/local/lib/python3.7/dist-packages/sklearn/utils/extmath.py:1021: RuntimeWarning: in
  new_unnormalized_variance -= correction ** 2 / new_sample_count
```

```
print(X)
```

```
[[-0.23640517  1.09706398 -2.07333501 ...  2.75062224  1.93701461
          nan]
 [-0.23640344  1.82982061 -0.35363241 ... -0.24388967  0.28118999
          nan]
 [ 0.43174109  1.57988811  0.45618695 ...  1.152255    0.20139121
          nan]
 ...
 [-0.23572747  0.70228425  2.0455738  ... -1.10454895 -0.31840916
          nan]
 [-0.23572517  1.83834103  2.33645719 ...  1.91908301  2.21963528
          nan]
 [-0.24240586 -1.80840125  1.22179204 ... -0.04813821 -0.75120669
          nan]]
```

Splitting the dataset into the Training set and Test set

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 0)
```

Part 2 - Building the ANN

Initializing the ANN

```
ann = tf.keras.models.Sequential()
```

Adding the input layer and the first hidden layer

```
ann.add(tf.keras.layers.Dense(units=6, activation='relu'))
```

Adding the second hidden layer

```
ann.add(tf.keras.layers.Dense(units=6, activation='relu'))
```

Adding the output layer

```
ann.add(tf.keras.layers.Dense(units=1, activation='sigmoid'))
```

Training the ANN

Compiling the ANN

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

Training the ANN on the Training set

```
ann.fit(X_train, y_train, batch_size = 32, epochs = 100)
```

```
Epoch 1/100
15/15 [==============================] - 1s 2ms/step - loss: nan - accuracy: 0.6374
Epoch 2/100
15/15 [==============================] - 0s 3ms/step - loss: nan - accuracy: 0.6374
Epoch 3/100
15/15 [==============================] - 0s 2ms/step - loss: nan - accuracy: 0.6374
Epoch 4/100
15/15 [==============================] - 0s 2ms/step - loss: nan - accuracy: 0.6374
Epoch 5/100
15/15 [==============================] - 0s 2ms/step - loss: nan - accuracy: 0.6374
Epoch 6/100
15/15 [==============================] - 0s 2ms/step - loss: nan - accuracy: 0.6374
Epoch 7/100
15/15 [==============================] - 0s 2ms/step - loss: nan - accuracy: 0.6374
Epoch 8/100
15/15 [==============================] - 0s 2ms/step - loss: nan - accuracy: 0.6374
```

```
Epoch 9/100
15/15 [==============================] - 0s 2ms/step - loss: nan - accuracy: 0.6374
Epoch 10/100
15/15 [==============================] - 0s 2ms/step - loss: nan - accuracy: 0.6374
Epoch 11/100
15/15 [==============================] - 0s 2ms/step - loss: nan - accuracy: 0.6374
Epoch 12/100
15/15 [==============================] - 0s 2ms/step - loss: nan - accuracy: 0.6374
Epoch 13/100
15/15 [==============================] - 0s 2ms/step - loss: nan - accuracy: 0.6374
Epoch 14/100
15/15 [==============================] - 0s 2ms/step - loss: nan - accuracy: 0.6374
Epoch 15/100
15/15 [==============================] - 0s 2ms/step - loss: nan - accuracy: 0.6374
Epoch 16/100
15/15 [==============================] - 0s 3ms/step - loss: nan - accuracy: 0.6374
Epoch 17/100
15/15 [==============================] - 0s 2ms/step - loss: nan - accuracy: 0.6374
Epoch 18/100
15/15 [==============================] - 0s 2ms/step - loss: nan - accuracy: 0.6374
Epoch 19/100
15/15 [==============================] - 0s 2ms/step - loss: nan - accuracy: 0.6374
Epoch 20/100
15/15 [==============================] - 0s 2ms/step - loss: nan - accuracy: 0.6374
Epoch 21/100
15/15 [==============================] - 0s 2ms/step - loss: nan - accuracy: 0.6374
Epoch 22/100
15/15 [==============================] - 0s 2ms/step - loss: nan - accuracy: 0.6374
Epoch 23/100
15/15 [==============================] - 0s 3ms/step - loss: nan - accuracy: 0.6374
Epoch 24/100
15/15 [==============================] - 0s 3ms/step - loss: nan - accuracy: 0.6374
Epoch 25/100
15/15 [==============================] - 0s 2ms/step - loss: nan - accuracy: 0.6374
Epoch 26/100
15/15 [==============================] - 0s 2ms/step - loss: nan - accuracy: 0.6374
Epoch 27/100
15/15 [==============================] - 0s 2ms/step - loss: nan - accuracy: 0.6374
Epoch 28/100
15/15 [==============================] - 0s 4ms/step - loss: nan - accuracy: 0.6374
Epoch 29/100
15/15 [==============================] - 0s 3ms/step - loss: nan - accuracy: 0.6374
```

Part 4 - Making the predictions and evaluating the model

Predicting the Test set results

```
y_pred = ann.predict(X_test)
y_pred = (y_pred > 0.5)
#print(np.concatenate((y_pred.reshape(len(y_pred),1), y_test.reshape(len(y_test),1)),1))
```

Making the Confusion Matrix

```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
print(cm)
```

```
[[67  0]
 [47  0]]
```