ANN Theory

Artifical Netural Network

In [1]:
```
pip install torch
```

```
Requirement already satisfied: torch in c:\users\roy62\anaconda3\envs\tensorflow_
env\lib\site-packages (2.5.1)
Requirement already satisfied: filelock in c:\users\roy62\anaconda3\envs\tensorfl
ow_env\lib\site-packages (from torch) (3.16.1)
Requirement already satisfied: typing-extensions>=4.8.0 in c:\users\roy62\anacond
a3\envs\tensorflow_env\lib\site-packages (from torch) (4.11.0)
Requirement already satisfied: networkx in c:\users\roy62\anaconda3\envs\tensorfl
ow_env\lib\site-packages (from torch) (3.4.2)
Requirement already satisfied: jinja2 in c:\users\roy62\anaconda3\envs\tensorflow
_env\lib\site-packages (from torch) (3.1.4)
Requirement already satisfied: fsspec in c:\users\roy62\anaconda3\envs\tensorflow
_env\lib\site-packages (from torch) (2024.10.0)
Requirement already satisfied: sympy==1.13.1 in c:\users\roy62\anaconda3\envs\ten
sorflow_env\lib\site-packages (from torch) (1.13.1)
Requirement already satisfied: mpmath<1.4,>=1.1.0 in c:\users\roy62\anaconda3\env
s\tensorflow_env\lib\site-packages (from sympy==1.13.1->torch) (1.3.0)
Requirement already satisfied: MarkupSafe>=2.0 in c:\users\roy62\anaconda3\envs\t
ensorflow_env\lib\site-packages (from jinja2->torch) (2.1.3)
Note: you may need to restart the kernel to use updated packages.
```

In [2]:
```
import numpy as np
```

In [3]:
```
import pandas as pd
```

In [4]:
```
import tensorflow as tf
```

In [8]:
```
tf.__version__
```

Out[8]:
```
'2.10.0'
```

In [9]:
```
# Part-1 Data Preprocessing

dataset = pd.read_csv(r"E:\Data Science & AI\Dataset files\Churn_Modelling.csv")
X = dataset.iloc[:, 3:-1].values
y = dataset.iloc[:, -1].values
```

In [10]:
```
print(X)
```

```
[[619 'delhi' 'Female' ... 1 1 101348.88]
 [608 'bangalore' 'Female' ... 0 1 112542.58]
 [502 'delhi' 'Female' ... 1 0 113931.57]
 ...
 [709 'delhi' 'Female' ... 0 1 42085.58]
 [772 'mumbai' 'Male' ... 1 0 92888.52]
 [792 'delhi' 'Female' ... 1 0 38190.78]]
```

In [11]:
```
print(y)
```

```
[1 0 1 ... 1 1 0]
```

In [12]:

```
# Encoding Categorical Dat
    ## Label Encoding Gender Column
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
X[:, 2] = le.fit_transform(X[:, 2])
```

In [13]:

```
print(X)
```

```
[[619 'delhi' 0 ... 1 1 101348.88]
 [608 'bangalore' 0 ... 0 1 112542.58]
 [502 'delhi' 0 ... 1 0 113931.57]
 ...
 [709 'delhi' 0 ... 0 1 42085.58]
 [772 'mumbai' 1 ... 1 0 92888.52]
 [792 'delhi' 0 ... 1 0 38190.78]]
```

In [14]:

```
# Geography Column

from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder
ct = ColumnTransformer(transformers=[('encoder', OneHotEncoder(), [1])], remaind
X = np.array(ct.fit_transform(X))
```

In [15]:

```
print(X)
```

```
[[0.0 1.0 0.0 ... 1 1 101348.88]
 [1.0 0.0 0.0 ... 0 1 112542.58]
 [0.0 1.0 0.0 ... 1 0 113931.57]
 ...
 [0.0 1.0 0.0 ... 0 1 42085.58]
 [0.0 0.0 1.0 ... 1 0 92888.52]
 [0.0 1.0 0.0 ... 1 0 38190.78]]
```

In [16]:

```
# Feature Scaling

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

In [17]:

```
print(X)
```

```
[[-0.57380915  0.99720391 -0.57873591 ...  0.64609167  0.97024255
   0.02188649]
 [ 1.74273971 -1.00280393 -0.57873591 ... -1.54776799  0.97024255
   0.21653375]
 [-0.57380915  0.99720391 -0.57873591 ...  0.64609167 -1.03067011
   0.2406869 ]
 ...
 [-0.57380915  0.99720391 -0.57873591 ... -1.54776799  0.97024255
  -1.00864308]
 [-0.57380915 -1.00280393  1.72790383 ...  0.64609167 -1.03067011
  -0.12523071]
 [-0.57380915  0.99720391 -0.57873591 ...  0.64609167 -1.03067011
  -1.07636976]]
```

In [18]:

```
# 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, randc
```

In [19]:

```
# Part 2 - Building the ANN
    # Initializing the ANN

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

In [22]:

```
# Adding the input layer and the first hidden layer

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

In [23]:

```
# Adding the second hidden layer

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

In [24]:

```
# Adding the output layer

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

In [25]:

```
# part-3 Training
    ## Compiling the ANN

ann.compile(optimizer = 'adam', loss = 'binary_crossentropy', metrics = ['accura
```

In [26]:

```
# Training the ANN on the Training set

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

```
Epoch 1/100
250/250 [==============================] - 2s 3ms/step - loss: 0.7205 - accuracy:
0.5660
Epoch 2/100
250/250 [==============================] - 1s 2ms/step - loss: 0.5018 - accuracy:
0.7930
Epoch 3/100
250/250 [==============================] - 1s 2ms/step - loss: 0.4460 - accuracy:
0.8021
Epoch 4/100
250/250 [==============================] - 1s 2ms/step - loss: 0.4307 - accuracy:
0.8173
Epoch 5/100
250/250 [==============================] - 1s 2ms/step - loss: 0.4240 - accuracy:
0.8201
Epoch 6/100
250/250 [==============================] - 1s 2ms/step - loss: 0.4192 - accuracy:
0.8211
Epoch 7/100
250/250 [==============================] - 1s 2ms/step - loss: 0.4146 - accuracy:
0.8221
Epoch 8/100
250/250 [==============================] - 1s 2ms/step - loss: 0.4092 - accuracy:
0.8280
Epoch 9/100
250/250 [==============================] - 1s 2ms/step - loss: 0.4029 - accuracy:
0.8295
Epoch 10/100
250/250 [==============================] - 1s 2ms/step - loss: 0.3940 - accuracy:
0.8369
Epoch 11/100
250/250 [==============================] - 1s 2ms/step - loss: 0.3826 - accuracy:
0.8432
Epoch 12/100
250/250 [==============================] - 1s 2ms/step - loss: 0.3703 - accuracy:
0.8526
Epoch 13/100
250/250 [==============================] - 1s 2ms/step - loss: 0.3613 - accuracy:
0.8551
Epoch 14/100
250/250 [==============================] - 1s 2ms/step - loss: 0.3555 - accuracy:
0.8569
Epoch 15/100
250/250 [==============================] - 1s 2ms/step - loss: 0.3523 - accuracy:
0.8572
Epoch 16/100
250/250 [==============================] - 1s 2ms/step - loss: 0.3499 - accuracy:
0.8575
Epoch 17/100
250/250 [==============================] - 1s 2ms/step - loss: 0.3483 - accuracy:
0.8593
Epoch 18/100
250/250 [==============================] - 1s 2ms/step - loss: 0.3472 - accuracy:
0.8571
Epoch 19/100
250/250 [==============================] - 1s 2ms/step - loss: 0.3464 - accuracy:
0.8586
Epoch 20/100
250/250 [==============================] - 1s 2ms/step - loss: 0.3458 - accuracy:
0.8595
```

```
Epoch 21/100
250/250 [==============================] - 1s 2ms/step - loss: 0.3451 - accuracy:
0.8601
Epoch 22/100
250/250 [==============================] - 1s 2ms/step - loss: 0.3447 - accuracy:
0.8595
Epoch 23/100
250/250 [==============================] - 1s 2ms/step - loss: 0.3441 - accuracy:
0.8605
Epoch 24/100
250/250 [==============================] - 1s 2ms/step - loss: 0.3439 - accuracy:
0.8605
Epoch 25/100
250/250 [==============================] - 1s 2ms/step - loss: 0.3431 - accuracy:
0.8619
Epoch 26/100
250/250 [==============================] - 1s 2ms/step - loss: 0.3425 - accuracy:
0.8602
Epoch 27/100
250/250 [==============================] - 1s 2ms/step - loss: 0.3425 - accuracy:
0.8609
Epoch 28/100
250/250 [==============================] - 1s 2ms/step - loss: 0.3422 - accuracy:
0.8614
Epoch 29/100
250/250 [==============================] - 1s 2ms/step - loss: 0.3420 - accuracy:
0.8609
Epoch 30/100
250/250 [==============================] - 1s 2ms/step - loss: 0.3415 - accuracy:
0.8610
Epoch 31/100
250/250 [==============================] - 1s 2ms/step - loss: 0.3414 - accuracy:
0.8614
Epoch 32/100
250/250 [==============================] - 1s 2ms/step - loss: 0.3408 - accuracy:
0.8605
Epoch 33/100
250/250 [==============================] - 1s 2ms/step - loss: 0.3407 - accuracy:
0.8609
Epoch 34/100
250/250 [==============================] - 1s 2ms/step - loss: 0.3409 - accuracy:
0.8611
Epoch 35/100
250/250 [==============================] - 1s 2ms/step - loss: 0.3400 - accuracy:
0.8621
Epoch 36/100
250/250 [==============================] - 1s 2ms/step - loss: 0.3399 - accuracy:
0.8615
Epoch 37/100
250/250 [==============================] - 1s 2ms/step - loss: 0.3398 - accuracy:
0.8606
Epoch 38/100
250/250 [==============================] - 1s 2ms/step - loss: 0.3394 - accuracy:
0.8621
Epoch 39/100
250/250 [==============================] - 1s 2ms/step - loss: 0.3391 - accuracy:
0.8624
Epoch 40/100
250/250 [==============================] - 1s 3ms/step - loss: 0.3390 - accuracy:
0.8611
```

```
Epoch 41/100
250/250 [==============================] - 1s 2ms/step - loss: 0.3388 - accuracy:
0.8618
Epoch 42/100
250/250 [==============================] - 1s 2ms/step - loss: 0.3384 - accuracy:
0.8631
Epoch 43/100
250/250 [==============================] - 1s 2ms/step - loss: 0.3383 - accuracy:
0.8610
Epoch 44/100
250/250 [==============================] - 1s 2ms/step - loss: 0.3378 - accuracy:
0.8640
Epoch 45/100
250/250 [==============================] - 1s 2ms/step - loss: 0.3377 - accuracy:
0.8625
Epoch 46/100
250/250 [==============================] - 1s 2ms/step - loss: 0.3375 - accuracy:
0.8636
Epoch 47/100
250/250 [==============================] - 1s 2ms/step - loss: 0.3373 - accuracy:
0.8615
Epoch 48/100
250/250 [==============================] - 1s 2ms/step - loss: 0.3372 - accuracy:
0.8616
Epoch 49/100
250/250 [==============================] - 1s 2ms/step - loss: 0.3371 - accuracy:
0.8629
Epoch 50/100
250/250 [==============================] - 1s 2ms/step - loss: 0.3367 - accuracy:
0.8619
Epoch 51/100
250/250 [==============================] - 1s 2ms/step - loss: 0.3364 - accuracy:
0.8639
Epoch 52/100
250/250 [==============================] - 1s 2ms/step - loss: 0.3364 - accuracy:
0.8626
Epoch 53/100
250/250 [==============================] - 1s 2ms/step - loss: 0.3361 - accuracy:
0.8626
Epoch 54/100
250/250 [==============================] - 1s 2ms/step - loss: 0.3357 - accuracy:
0.8636
Epoch 55/100
250/250 [==============================] - 1s 2ms/step - loss: 0.3356 - accuracy:
0.8640
Epoch 56/100
250/250 [==============================] - 1s 2ms/step - loss: 0.3352 - accuracy:
0.8611
Epoch 57/100
250/250 [==============================] - 1s 2ms/step - loss: 0.3356 - accuracy:
0.8646
Epoch 58/100
250/250 [==============================] - 1s 2ms/step - loss: 0.3353 - accuracy:
0.8627
Epoch 59/100
250/250 [==============================] - 1s 2ms/step - loss: 0.3350 - accuracy:
0.8649
Epoch 60/100
250/250 [==============================] - 1s 2ms/step - loss: 0.3347 - accuracy:
0.8634
```

```
Epoch 61/100
250/250 [==============================] - 1s 2ms/step - loss: 0.3348 - accuracy:
0.8655
Epoch 62/100
250/250 [==============================] - 1s 2ms/step - loss: 0.3347 - accuracy:
0.8641
Epoch 63/100
250/250 [==============================] - 1s 2ms/step - loss: 0.3345 - accuracy:
0.8645
Epoch 64/100
250/250 [==============================] - 1s 2ms/step - loss: 0.3343 - accuracy:
0.8629
Epoch 65/100
250/250 [==============================] - 1s 2ms/step - loss: 0.3344 - accuracy:
0.8634
Epoch 66/100
250/250 [==============================] - 1s 2ms/step - loss: 0.3341 - accuracy:
0.8645
Epoch 67/100
250/250 [==============================] - 1s 2ms/step - loss: 0.3340 - accuracy:
0.8625
Epoch 68/100
250/250 [==============================] - 1s 2ms/step - loss: 0.3345 - accuracy:
0.8630
Epoch 69/100
250/250 [==============================] - 1s 2ms/step - loss: 0.3338 - accuracy:
0.8621
Epoch 70/100
250/250 [==============================] - 1s 2ms/step - loss: 0.3341 - accuracy:
0.8636
Epoch 71/100
250/250 [==============================] - 1s 2ms/step - loss: 0.3338 - accuracy:
0.8630
Epoch 72/100
250/250 [==============================] - 1s 2ms/step - loss: 0.3338 - accuracy:
0.8631
Epoch 73/100
250/250 [==============================] - 1s 2ms/step - loss: 0.3335 - accuracy:
0.8633
Epoch 74/100
250/250 [==============================] - 1s 2ms/step - loss: 0.3335 - accuracy:
0.8635
Epoch 75/100
250/250 [==============================] - 1s 2ms/step - loss: 0.3335 - accuracy:
0.8624
Epoch 76/100
250/250 [==============================] - 1s 2ms/step - loss: 0.3334 - accuracy:
0.8636
Epoch 77/100
250/250 [==============================] - 1s 2ms/step - loss: 0.3332 - accuracy:
0.8644
Epoch 78/100
250/250 [==============================] - 1s 2ms/step - loss: 0.3329 - accuracy:
0.8651
Epoch 79/100
250/250 [==============================] - 1s 2ms/step - loss: 0.3333 - accuracy:
0.8644
Epoch 80/100
250/250 [==============================] - 1s 2ms/step - loss: 0.3338 - accuracy:
0.8626
```

```
Epoch 81/100
250/250 [==============================] - 1s 2ms/step - loss: 0.3327 - accuracy:
0.8640
Epoch 82/100
250/250 [==============================] - 1s 2ms/step - loss: 0.3330 - accuracy:
0.8640
Epoch 83/100
250/250 [==============================] - 1s 2ms/step - loss: 0.3325 - accuracy:
0.8646
Epoch 84/100
250/250 [==============================] - 1s 2ms/step - loss: 0.3322 - accuracy:
0.8643
Epoch 85/100
250/250 [==============================] - 1s 2ms/step - loss: 0.3322 - accuracy:
0.8648
Epoch 86/100
250/250 [==============================] - 1s 2ms/step - loss: 0.3323 - accuracy:
0.8634
Epoch 87/100
250/250 [==============================] - 1s 2ms/step - loss: 0.3318 - accuracy:
0.8640
Epoch 88/100
250/250 [==============================] - 1s 2ms/step - loss: 0.3323 - accuracy:
0.8633
Epoch 89/100
250/250 [==============================] - 1s 2ms/step - loss: 0.3321 - accuracy:
0.8658
Epoch 90/100
250/250 [==============================] - 1s 2ms/step - loss: 0.3316 - accuracy:
0.8645
Epoch 91/100
250/250 [==============================] - 1s 2ms/step - loss: 0.3320 - accuracy:
0.8646
Epoch 92/100
250/250 [==============================] - 1s 2ms/step - loss: 0.3319 - accuracy:
0.8629
Epoch 93/100
250/250 [==============================] - 1s 2ms/step - loss: 0.3315 - accuracy:
0.8645
Epoch 94/100
250/250 [==============================] - 1s 2ms/step - loss: 0.3317 - accuracy:
0.8640
Epoch 95/100
250/250 [==============================] - 1s 2ms/step - loss: 0.3314 - accuracy:
0.8646
Epoch 96/100
250/250 [==============================] - 1s 2ms/step - loss: 0.3321 - accuracy:
0.8641
Epoch 97/100
250/250 [==============================] - 1s 2ms/step - loss: 0.3314 - accuracy:
0.8641
Epoch 98/100
250/250 [==============================] - 1s 2ms/step - loss: 0.3313 - accuracy:
0.8648
Epoch 99/100
250/250 [==============================] - 1s 2ms/step - loss: 0.3314 - accuracy:
0.8640
Epoch 100/100
250/250 [==============================] - 1s 2ms/step - loss: 0.3317 - accuracy:
0.8627
```

Out[26]:

```
<keras.callbacks.History at 0x239574cd750>
```

In [27]:

```python
# 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),
```

```
63/63 [==============================] - 0s 2ms/step
[[0 0]
 [0 1]
 [0 0]
 ...
 [0 0]
 [0 0]
 [0 0]]
```

In [28]:

```python
# Making the Confusion Matrix

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

```
[[1518   77]
 [ 197  208]]
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: