



In [1]:

```
1 import numpy as np
2 import pandas as pd
3
4 import tensorflow as tf
5 import keras
```

In [2]:

```
1 df = pd.read_csv('Churn_Modelling.csv')
2 df.head()
```

Out[2]:

|   | RowNumber | CustomerId | Surname  | CreditScore | Geography | Gender | Age | Tenure | Balance   | NumOfProducts | HasCrCard | IsActiveMe |
|---|-----------|------------|----------|-------------|-----------|--------|-----|--------|-----------|---------------|-----------|------------|
| 0 | 1         | 15634602   | Hargrave | 619         | France    | Female | 42  | 2      | 0.00      | 1             | 1         |            |
| 1 | 2         | 15647311   | Hill     | 608         | Spain     | Female | 41  | 1      | 83807.86  | 1             | 0         |            |
| 2 | 3         | 15619304   | Onio     | 502         | France    | Female | 42  | 8      | 159660.80 | 3             | 1         |            |
| 3 | 4         | 15701354   | Boni     | 699         | France    | Female | 39  | 1      | 0.00      | 2             | 0         |            |
| 4 | 5         | 15737888   | Mitchell | 850         | Spain     | Female | 43  | 2      | 125510.82 | 1             | 1         |            |

In [3]:

```
1 df["Exited"].value_counts()
```

Out[3]:

```
0    7963
1    2037
Name: Exited, dtype: int64
```



In [4]: 1 df.info()

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 14 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   RowNumber        10000 non-null   int64  
 1   CustomerId       10000 non-null   int64  
 2   Surname          10000 non-null   object  
 3   CreditScore      10000 non-null   int64  
 4   Geography         10000 non-null   object  
 5   Gender            10000 non-null   object  
 6   Age               10000 non-null   int64  
 7   Tenure            10000 non-null   int64  
 8   Balance           10000 non-null   float64 
 9   NumOfProducts     10000 non-null   int64  
 10  HasCrCard        10000 non-null   int64  
 11  IsActiveMember    10000 non-null   int64  
 12  EstimatedSalary   10000 non-null   float64 
 13  Exited            10000 non-null   int64  
dtypes: float64(2), int64(9), object(3)
memory usage: 1.1+ MB
```

In [5]: 1 df.isnull().sum()

Out[5]:

|                 |   |
|-----------------|---|
| RowNumber       | 0 |
| CustomerId      | 0 |
| Surname         | 0 |
| CreditScore     | 0 |
| Geography       | 0 |
| Gender          | 0 |
| Age             | 0 |
| Tenure          | 0 |
| Balance         | 0 |
| NumOfProducts   | 0 |
| HasCrCard       | 0 |
| IsActiveMember  | 0 |
| EstimatedSalary | 0 |
| Exited          | 0 |
| dtype: int64    |   |



In [6]:

```
1 df.drop(columns=["RowNumber", "CustomerId", "Surname"], inplace=True)
2 df
```

Out[6]:

|      | CreditScore | Geography | Gender | Age | Tenure | Balance   | NumOfProducts | HasCrCard | IsActiveMember | EstimatedSalary | Exited |
|------|-------------|-----------|--------|-----|--------|-----------|---------------|-----------|----------------|-----------------|--------|
| 0    | 619         | France    | Female | 42  | 2      | 0.00      | 1             | 1         | 1              | 101348.88       | 1      |
| 1    | 608         | Spain     | Female | 41  | 1      | 83807.86  | 1             | 0         | 1              | 112542.58       | 0      |
| 2    | 502         | France    | Female | 42  | 8      | 159660.80 | 3             | 1         | 0              | 113931.57       | 1      |
| 3    | 699         | France    | Female | 39  | 1      | 0.00      | 2             | 0         | 0              | 93826.63        | 0      |
| 4    | 850         | Spain     | Female | 43  | 2      | 125510.82 | 1             | 1         | 1              | 79084.10        | 0      |
| ...  | ...         | ...       | ...    | ... | ...    | ...       | ...           | ...       | ...            | ...             | ...    |
| 9995 | 771         | France    | Male   | 39  | 5      | 0.00      | 2             | 1         | 0              | 96270.64        | 0      |
| 9996 | 516         | France    | Male   | 35  | 10     | 57369.61  | 1             | 1         | 1              | 101699.77       | 0      |
| 9997 | 709         | France    | Female | 36  | 7      | 0.00      | 1             | 0         | 1              | 42085.58        | 1      |
| 9998 | 772         | Germany   | Male   | 42  | 3      | 75075.31  | 2             | 1         | 0              | 92888.52        | 1      |
| 9999 | 792         | France    | Female | 28  | 4      | 130142.79 | 1             | 1         | 0              | 38190.78        | 0      |

10000 rows x 11 columns

In [7]:

```
1 df['Geography'].value_counts()
```

Out[7]:

```
France    5014
Germany   2509
Spain     2477
Name: Geography, dtype: int64
```

In [8]:

```
1 df['Gender'].value_counts()
```

Out[8]:

```
Male      5457
Female    4543
Name: Gender, dtype: int64
```

In [9]:

```
1 X = pd.get_dummies(df.drop('Exited', axis=1), drop_first=True)
2 y = df['Exited']
```



In [10]: 1 X

Out[10]:

|      | CreditScore | Age | Tenure | Balance   | NumOfProducts | HasCrCard | IsActiveMember | EstimatedSalary | Geography_Germany | Geograph |
|------|-------------|-----|--------|-----------|---------------|-----------|----------------|-----------------|-------------------|----------|
| 0    | 619         | 42  | 2      | 0.00      |               | 1         | 1              | 1               | 101348.88         | 0        |
| 1    | 608         | 41  | 1      | 83807.86  |               | 1         | 0              | 1               | 112542.58         | 0        |
| 2    | 502         | 42  | 8      | 159660.80 |               | 3         | 1              | 0               | 113931.57         | 0        |
| 3    | 699         | 39  | 1      | 0.00      |               | 2         | 0              | 0               | 93826.63          | 0        |
| 4    | 850         | 43  | 2      | 125510.82 |               | 1         | 1              | 1               | 79084.10          | 0        |
| ...  | ...         | ... | ...    | ...       |               | ...       | ...            | ...             | ...               | ...      |
| 9995 | 771         | 39  | 5      | 0.00      |               | 2         | 1              | 0               | 96270.64          | 0        |
| 9996 | 516         | 35  | 10     | 57369.61  |               | 1         | 1              | 1               | 101699.77         | 0        |
| 9997 | 709         | 36  | 7      | 0.00      |               | 1         | 0              | 1               | 42085.58          | 0        |
| 9998 | 772         | 42  | 3      | 75075.31  |               | 2         | 1              | 0               | 92888.52          | 1        |
| 9999 | 792         | 28  | 4      | 130142.79 |               | 1         | 1              | 0               | 38190.78          | 0        |

10000 rows × 11 columns

In [11]:

```
1 ### Splitting the dataset into the Training set and Test set
2 from sklearn.model_selection import train_test_split
3 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 0)
```

In [12]:

```
1 X_train.shape,X_test.shape,y_train.shape,y_test.shape
```

Out[12]:

```
((8000, 11), (2000, 11), (8000,), (2000,))
```

In [13]:

```
1 from sklearn.preprocessing import StandardScaler
2 sc = StandardScaler()
3 X_train = sc.fit_transform(X_train)
4 X_test = sc.transform(X_test)
```

## Modelling



## Initializing the Artificial Neural Network

```
In [14]: 1 from keras.models import Sequential  
2 ann = Sequential()
```

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

```
In [15]: 1 from keras.layers import Dense  
2 ann.add(Dense(input_dim = 11,units = 6, kernel_initializer = 'uniform', activation = 'relu'))  
3  
4 #ann.add(Dropout(rate = 0.1))
```

### Adding the second hidden layer

```
In [16]: 1 ann.add(Dense(units = 6, kernel_initializer = 'uniform', activation = 'relu'))  
2  
3 #ann.add(Dropout(rate = 0.1))
```

### Adding the output layer

```
In [17]: 1 ann.add(Dense(units = 1, kernel_initializer = 'uniform', activation = 'sigmoid'))
```

### Compiling the ANN

```
In [18]: 1 ann.compile(optimizer = 'adam', loss = 'binary_crossentropy', metrics = ['accuracy'])
```

### Training the ANN on the Training set



```
In [19]: 1 ann.fit(X_train, y_train, epochs = 100,batch_size=32)
          250/250 [=====] - 0s 1ms/step - loss: 0.3399 - accuracy: 0.8615
          Epoch 60/100
          250/250 [=====] - 0s 1ms/step - loss: 0.3401 - accuracy: 0.8614
          Epoch 61/100
          250/250 [=====] - 0s 1ms/step - loss: 0.3400 - accuracy: 0.8601
          Epoch 62/100
          250/250 [=====] - 0s 1ms/step - loss: 0.3405 - accuracy: 0.8610
          Epoch 63/100
          250/250 [=====] - 0s 1ms/step - loss: 0.3396 - accuracy: 0.8627
          Epoch 64/100
          250/250 [=====] - 0s 1ms/step - loss: 0.3394 - accuracy: 0.8594
          Epoch 65/100
          250/250 [=====] - 0s 1ms/step - loss: 0.3402 - accuracy: 0.8606
          Epoch 66/100
          250/250 [=====] - 0s 1ms/step - loss: 0.3388 - accuracy: 0.8612
          Epoch 67/100
          250/250 [=====] - 0s 1ms/step - loss: 0.3397 - accuracy: 0.8633
          Epoch 68/100
          250/250 [=====] - 0s 1ms/step - loss: 0.3397 - accuracy: 0.8627
```

## Predictions & Evaluating the model

### Predictions

```
In [20]: 1 y_pred = ann.predict(X_test)
          2 y_pred = (y_pred > 0.5)
          3
          4
          5
          6
          7
          8
          9
          10
          11
          12
          13
          14
          15
          16
          17
          18
          19
          20
          21
          22
          23
          24
          25
          26
          27
          28
          29
          30
          31
          32
          33
          34
          35
          36
          37
          38
          39
          40
          41
          42
          43
          44
          45
          46
          47
          48
          49
          50
          51
          52
          53
          54
          55
          56
          57
          58
          59
          60
          61
          62
          63
          64
          65
          66
          67
          68
          69
          70
          71
          72
          73
          74
          75
          76
          77
          78
          79
          80
          81
          82
          83
          84
          85
          86
          87
          88
          89
          90
          91
          92
          93
          94
          95
          96
          97
          98
          99
          100
          101
          102
          103
          104
          105
          106
          107
          108
          109
          110
          111
          112
          113
          114
          115
          116
          117
          118
          119
          120
          121
          122
          123
          124
          125
          126
          127
          128
          129
          130
          131
          132
          133
          134
          135
          136
          137
          138
          139
          140
          141
          142
          143
          144
          145
          146
          147
          148
          149
          150
          151
          152
          153
          154
          155
          156
          157
          158
          159
          160
          161
          162
          163
          164
          165
          166
          167
          168
          169
          170
          171
          172
          173
          174
          175
          176
          177
          178
          179
          180
          181
          182
          183
          184
          185
          186
          187
          188
          189
          190
          191
          192
          193
          194
          195
          196
          197
          198
          199
          200
          201
          202
          203
          204
          205
          206
          207
          208
          209
          210
          211
          212
          213
          214
          215
          216
          217
          218
          219
          220
          221
          222
          223
          224
          225
          226
          227
          228
          229
          230
          231
          232
          233
          234
          235
          236
          237
          238
          239
          240
          241
          242
          243
          244
          245
          246
          247
          248
          249
          250
          251
          252
          253
          254
          255
          256
          257
          258
          259
          260
          261
          262
          263
          264
          265
          266
          267
          268
          269
          270
          271
          272
          273
          274
          275
          276
          277
          278
          279
          280
          281
          282
          283
          284
          285
          286
          287
          288
          289
          290
          291
          292
          293
          294
          295
          296
          297
          298
          299
          300
          301
          302
          303
          304
          305
          306
          307
          308
          309
          310
          311
          312
          313
          314
          315
          316
          317
          318
          319
          320
          321
          322
          323
          324
          325
          326
          327
          328
          329
          330
          331
          332
          333
          334
          335
          336
          337
          338
          339
          340
          341
          342
          343
          344
          345
          346
          347
          348
          349
          350
          351
          352
          353
          354
          355
          356
          357
          358
          359
          360
          361
          362
          363
          364
          365
          366
          367
          368
          369
          370
          371
          372
          373
          374
          375
          376
          377
          378
          379
          380
          381
          382
          383
          384
          385
          386
          387
          388
          389
          390
          391
          392
          393
          394
          395
          396
          397
          398
          399
          400
          401
          402
          403
          404
          405
          406
          407
          408
          409
          410
          411
          412
          413
          414
          415
          416
          417
          418
          419
          420
          421
          422
          423
          424
          425
          426
          427
          428
          429
          430
          431
          432
          433
          434
          435
          436
          437
          438
          439
          440
          441
          442
          443
          444
          445
          446
          447
          448
          449
          450
          451
          452
          453
          454
          455
          456
          457
          458
          459
          460
          461
          462
          463
          464
          465
          466
          467
          468
          469
          470
          471
          472
          473
          474
          475
          476
          477
          478
          479
          480
          481
          482
          483
          484
          485
          486
          487
          488
          489
          490
          491
          492
          493
          494
          495
          496
          497
          498
          499
          500
          501
          502
          503
          504
          505
          506
          507
          508
          509
          510
          511
          512
          513
          514
          515
          516
          517
          518
          519
          520
          521
          522
          523
          524
          525
          526
          527
          528
          529
          530
          531
          532
          533
          534
          535
          536
          537
          538
          539
          540
          541
          542
          543
          544
          545
          546
          547
          548
          549
          550
          551
          552
          553
          554
          555
          556
          557
          558
          559
          560
          561
          562
          563
          564
          565
          566
          567
          568
          569
          570
          571
          572
          573
          574
          575
          576
          577
          578
          579
          580
          581
          582
          583
          584
          585
          586
          587
          588
          589
          590
          591
          592
          593
          594
          595
          596
          597
          598
          599
          600
          601
          602
          603
          604
          605
          606
          607
          608
          609
          610
          611
          612
          613
          614
          615
          616
          617
          618
          619
          620
          621
          622
          623
          624
          625
          626
          627
          628
          629
          630
          631
          632
          633
          634
          635
          636
          637
          638
          639
          640
          641
          642
          643
          644
          645
          646
          647
          648
          649
          650
          651
          652
          653
          654
          655
          656
          657
          658
          659
          660
          661
          662
          663
          664
          665
          666
          667
          668
          669
          670
          671
          672
          673
          674
          675
          676
          677
          678
          679
          680
          681
          682
          683
          684
          685
          686
          687
          688
          689
          690
          691
          692
          693
          694
          695
          696
          697
          698
          699
          700
          701
          702
          703
          704
          705
          706
          707
          708
          709
          710
          711
          712
          713
          714
          715
          716
          717
          718
          719
          720
          721
          722
          723
          724
          725
          726
          727
          728
          729
          730
          731
          732
          733
          734
          735
          736
          737
          738
          739
          740
          741
          742
          743
          744
          745
          746
          747
          748
          749
          750
          751
          752
          753
          754
          755
          756
          757
          758
          759
          760
          761
          762
          763
          764
          765
          766
          767
          768
          769
          770
          771
          772
          773
          774
          775
          776
          777
          778
          779
          780
          781
          782
          783
          784
          785
          786
          787
          788
          789
          790
          791
          792
          793
          794
          795
          796
          797
          798
          799
          800
          801
          802
          803
          804
          805
          806
          807
          808
          809
          8010
          8011
          8012
          8013
          8014
          8015
          8016
          8017
          8018
          8019
          8020
          8021
          8022
          8023
          8024
          8025
          8026
          8027
          8028
          8029
          8030
          8031
          8032
          8033
          8034
          8035
          8036
          8037
          8038
          8039
          80310
          80311
          80312
          80313
          80314
          80315
          80316
          80317
          80318
          80319
          80320
          80321
          80322
          80323
          80324
          80325
          80326
          80327
          80328
          80329
          80330
          80331
          80332
          80333
          80334
          80335
          80336
          80337
          80338
          80339
          80340
          80341
          80342
          80343
          80344
          80345
          80346
          80347
          80348
          80349
          80350
          80351
          80352
          80353
          80354
          80355
          80356
          80357
          80358
          80359
          80360
          80361
          80362
          80363
          80364
          80365
          80366
          80367
          80368
          80369
          80370
          80371
          80372
          80373
          80374
          80375
          80376
          80377
          80378
          80379
          80380
          80381
          80382
          80383
          80384
          80385
          80386
          80387
          80388
          80389
          80390
          80391
          80392
          80393
          80394
          80395
          80396
          80397
          80398
          80399
          803100
          803101
          803102
          803103
          803104
          803105
          803106
          803107
          803108
          803109
          803110
          803111
          803112
          803113
          803114
          803115
          803116
          803117
          803118
          803119
          803120
          803121
          803122
          803123
          803124
          803125
          803126
          803127
          803128
          803129
          803130
          803131
          803132
          803133
          803134
          803135
          803136
          803137
          803138
          803139
          803140
          803141
          803142
          803143
          803144
          803145
          803146
          803147
          803148
          803149
          803150
          803151
          803152
          803153
          803154
          803155
          803156
          803157
          803158
          803159
          803160
          803161
          803162
          803163
          803164
          803165
          803166
          803167
          803168
          803169
          803170
          803171
          803172
          803173
          803174
          803175
          803176
          803177
          803178
          803179
          803180
          803181
          803182
          803183
          803184
          803185
          803186
          803187
          803188
          803189
          803190
          803191
          803192
          803193
          803194
          803195
          803196
          803197
          803198
          803199
          803200
          803201
          803202
          803203
          803204
          803205
          803206
          803207
          803208
          803209
          803210
          803211
          803212
          803213
          803214
          803215
          803216
          803217
          803218
          803219
          803220
          803221
          803222
          803223
          803224
          803225
          803226
          803227
          803228
          803229
          803230
          803231
          803232
          803233
          803234
          803235
          803236
          803237
          803238
          803239
          803240
          803241
          803242
          803243
          803244
          803245
          803246
          803247
          803248
          803249
          803250
          803251
          803252
          803253
          803254
          803255
          803256
          803257
          803258
          803259
          803260
          803261
          803262
          803263
          803264
          803265
          803266
          803267
          803268
          803269
          803270
          803271
          803272
          803273
          803274
          803275
          803276
          803277
          803278
          803279
          803280
          803281
          803282
          803283
          803284
          803285
          803286
          803287
          803288
          803289
          803290
          803291
          803292
          803293
          803294
          803295
          803296
          803297
          803298
          803299
          803300
          803301
          803302
          803303
          803304
          803305
          803306
          803307
          803308
          803309
          803310
          803311
          803312
          803313
          803314
          803315
          803316
          803317
          803318
          803319
          803320
          803321
          803322
          803323
          803324
          803325
          803326
          803327
          803328
          803329
          803330
          803331
          803332
          803333
          803334
          803335
          803336
          803337
          803338
          803339
          8033310
          8033311
          8033312
          8033313
          8033314
          8033315
          8033316
          8033317
          8033318
          8033319
          80333110
          80333111
          80333112
          80333113
          80333114
          80333115
          80333116
          80333117
          80333118
          80333119
          803331110
          803331111
          803331112
          803331113
          803331114
          803331115
          803331116
          803331117
          803331118
          803331119
          8033311110
          8033311111
          8033311112
          8033311113
          8033311114
          8033311115
          8033311116
          8033311117
          8033311118
          8033311119
          80333111110
          80333111111
          80333111112
          80333111113
          80333111114
          80333111115
          80333111116
          80333111117
          80333111118
          80333111119
          803331111110
          803331111111
          803331111112
          803331111113
          803331111114
          803331111115
          803331111116
          803331111117
          803331111118
          803331111119
          8033311111110
          8033311111111
          8033311111112
          8033311111113
          8033311111114
          8033311111115
          8033311111116
          8033311111117
          8033311111118
          8033311111119
          80333111111110
          80333111111111
          80333111111112
          80333111111113
          80333111111114
          80333111111115
          80333111111116
          80333111111117
          80333111111118
          80333111111119
          803331111111110
          803331111111111
          803331111111112
          803331111111113
          803331111111114
          803331111111115
          803331111111116
          803331111111117
          803331111111118
          803331111111119
          8033311111111110
          8033311111111111
          8033311111111112
          8033311111111113
          8033311111111114
          8033311111111115
          8033311111111116
          8033311111111117
          8033311111111118
          8033311111111119
          80333111111111110
          80333111111111111
          80333111111111112
          80333111111111113
          80333111111111114
          80333111111111115
          80333111111111116
          80333111111111117
          80333111111111118
          80333111111111119
          803331111111111110
          803331111111111111
          803331111111111112
          803331111111111113
          803331111111111114
          803331111111111115
          803331111111111116
          803331111111111117
          803331111111111118
          803331111111111119
          8033311111111111110
          8033311111111111111
          8033311111111111112
          8033311111111111113
          8033311111111111114
          8033311111111111115
          8033311111111111116
          8033311111111111117
          8033311111111111118
          8033311111111111119
          80333111111111111110
          80333111111111111111
          80333111111111111112
          80333111111111111113
          80333111111111111114
          80333111111111111115
          80333111111111111116
          80333111111111111117
          80333111111111111118
          80333111111111111119
          803331111111111111110
          803331111111111111111
          803331111111111111112
          803331111111111111113
          803331111111111111114
          803331111111111111115
          803331111111111111116
          803331111111111111117
          803331111111111111118
          803331111111111111119
          8033311111111111111110
          8033311111111111111111
          8033311111111111111112
          8033311111111111111113
          8033311111111111111114
          8033311111111111111115
          803331111111111111111
```



```
In [21]: 1 from sklearn.metrics import confusion_matrix, accuracy_score  
2  
3 print("Test Accuracy:",accuracy_score(y_test, y_pred))  
4 confusion_matrix(y_test, y_pred)
```

Test Accuracy: 0.8555

```
Out[21]: array([[1501,    94],  
                 [ 195,   210]], dtype=int64)
```

## Cross validate the model

```
In [22]: 1 def build_cross_classifier():  
2     classifier = Sequential()  
3     classifier.add(Dense(input_dim = 11, units = 6, kernel_initializer = 'uniform', activation = 'relu'))  
4     classifier.add(Dense(units = 6, kernel_initializer = 'uniform', activation = 'relu'))  
5     classifier.add(Dense(units = 1, kernel_initializer = 'uniform', activation = 'sigmoid'))  
6     classifier.compile(optimizer = "adam", loss = 'binary_crossentropy', metrics = ['accuracy'])  
7     return classifier
```

```
In [23]: 1 from scikeras.wrappers import KerasClassifier  
2 classifier = KerasClassifier(build_cross_classifier,batch_size=32,epochs=100)
```



In [24]:

```
1 from sklearn.model_selection import cross_val_score  
2 accuracies = cross_val_score(classifier, X, y, cv = 5)
```

```
Epoch 1/100  
250/250 [=====] - 1s 1ms/step - loss: 0.5371 - accuracy: 0.7937  
Epoch 2/100  
250/250 [=====] - 0s 1ms/step - loss: 0.5236 - accuracy: 0.7964  
Epoch 3/100  
250/250 [=====] - 0s 2ms/step - loss: 0.5169 - accuracy: 0.7964  
Epoch 4/100  
250/250 [=====] - 0s 1ms/step - loss: 0.5104 - accuracy: 0.7964  
Epoch 5/100  
250/250 [=====] - 0s 1ms/step - loss: 0.5076 - accuracy: 0.7964  
Epoch 6/100  
250/250 [=====] - 0s 2ms/step - loss: 0.5083 - accuracy: 0.7964  
Epoch 7/100  
250/250 [=====] - 0s 1ms/step - loss: 0.5044 - accuracy: 0.7964  
Epoch 8/100  
250/250 [=====] - 0s 1ms/step - loss: 0.5062 - accuracy: 0.7964  
Epoch 9/100  
250/250 [=====] - 0s 1ms/step - loss: 0.5055 - accuracy: 0.7964  
Epoch 10/100  
250/250 [=====] - 0s 1ms/step - loss: 0.5055 - accuracy: 0.7964
```

In [25]:

```
1 print(accuracies)  
2 accuracies.mean()
```

```
[0.796 0.796 0.7965 0.7965 0.7965]
```

Out[25]: 0.7963

## Improving and Tuning the ANN

This can be done by using 3 options

- 1.Hyperparameter Tuning
- 2.Regularization (L1 & L2) to reduce overfitting if needed (for Regression problems)
- 3.Dropout

Hyperparameter tuning can be done for identifying no.of hidden layers & no.of neurons in each hidden layer



- layers = [[20],[10],[30],[40]] --> 1 hidden layer with different options of no.of neurons in that hidden layer
- layers = [[20],[40,20],[45,30,15]] --> Multiple hidden layers with different options of no.of neurons

**Hyperparameter tuning can be done for identifying best activation function for hidden layers**

- activations=['relu','sigmoid']

**Hyperparameter tuning can be done for identifying best optimizers**

- optimizer =['adam','rmsprop']

**Hyperparameter tuning can be done for identifying best batchsize for building/training model**

- batchsize =[10,16,32,64,128,256]

```
In [27]: 1 estimator = KerasClassifier(build_cross_classifier())
          2
          3 param_grid = {'batch_size': [10,32], 'epochs': [50,100],"optimizer" :['adam','rmsprop']}
```

```
In [28]: 1 from sklearn.model_selection import GridSearchCV
          2
          3 grid = GridSearchCV(estimator, param_grid, scoring = 'accuracy', cv = 5)
          4
          5 grid_result = grid.fit(X_train, y_train)
```

...

```
In [29]: 1 #best parameters
          2 grid_result.best_params_
```

```
Out[29]: {'batch_size': 10, 'epochs': 100, 'optimizer': 'rmsprop'}
```

```
In [30]: 1 #best accuracy
          2 grid_result.best_score_
```

```
Out[30]: 0.8428749999999999
```