Objective:::DL Model for Bank loan Approval[¶](#gjdgxs)

In [2]:

**import** pandas **as** pd  
**import** numpy **as** np  
**import** matplotlib.pyplot **as** plt

Data Collection/Extraction[¶](#30j0zll)

In [3]:

df**=**pd**.**read\_csv (r"C:\Users\star\Desktop\DeepLearningWithPython\GoogleDrive\assignment\assignment -2\Loan-Approval-Prediction.csv")

Data Exploration[¶](#1fob9te)

In [4]:

df**.**head(2)

Out[4]:

|  | **Loan\_ID** | **Gender** | **Married** | **Dependents** | **Education** | **Self\_Employed** | **ApplicantIncome** | **CoapplicantIncome** | **LoanAmount** | **Loan\_Amount\_Term** | **Credit\_History** | **Property\_Area** | **Loan\_Status** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | LP001002 | Male | No | 0 | Graduate | No | 5849 | 0.0 | NaN | 360.0 | 1.0 | Urban | Y |
| **1** | LP001003 | Male | Yes | 1 | Graduate | No | 4583 | 1508.0 | 128.0 | 360.0 | 1.0 | Rural | N |

In [5]:

df**.**describe()

Out[5]:

|  | **ApplicantIncome** | **CoapplicantIncome** | **LoanAmount** | **Loan\_Amount\_Term** | **Credit\_History** |
| --- | --- | --- | --- | --- | --- |
| **count** | 614.000000 | 614.000000 | 592.000000 | 600.00000 | 564.000000 |
| **mean** | 5403.459283 | 1621.245798 | 146.412162 | 342.00000 | 0.842199 |
| **std** | 6109.041673 | 2926.248369 | 85.587325 | 65.12041 | 0.364878 |
| **min** | 150.000000 | 0.000000 | 9.000000 | 12.00000 | 0.000000 |
| **25%** | 2877.500000 | 0.000000 | 100.000000 | 360.00000 | 1.000000 |
| **50%** | 3812.500000 | 1188.500000 | 128.000000 | 360.00000 | 1.000000 |
| **75%** | 5795.000000 | 2297.250000 | 168.000000 | 360.00000 | 1.000000 |
| **max** | 81000.000000 | 41667.000000 | 700.000000 | 480.00000 | 1.000000 |

In [6]:

df**.**shape

Out[6]:

(614, 13)

In [7]:

df**.**size

Out[7]:

7982

In [8]:

df**.**info()

<class 'pandas.core.frame.DataFrame'>  
RangeIndex: 614 entries, 0 to 613  
Data columns (total 13 columns):  
 # Column Non-Null Count Dtype   
--- ------ -------------- -----   
 0 Loan\_ID 614 non-null object   
 1 Gender 601 non-null object   
 2 Married 611 non-null object   
 3 Dependents 599 non-null object   
 4 Education 614 non-null object   
 5 Self\_Employed 582 non-null object   
 6 ApplicantIncome 614 non-null int64   
 7 CoapplicantIncome 614 non-null float64  
 8 LoanAmount 592 non-null float64  
 9 Loan\_Amount\_Term 600 non-null float64  
 10 Credit\_History 564 non-null float64  
 11 Property\_Area 614 non-null object   
 12 Loan\_Status 614 non-null object   
dtypes: float64(4), int64(1), object(8)  
memory usage: 62.5+ KB

In [9]:

df**.**isnull()**.**sum()

Out[9]:

Loan\_ID 0  
Gender 13  
Married 3  
Dependents 15  
Education 0  
Self\_Employed 32  
ApplicantIncome 0  
CoapplicantIncome 0  
LoanAmount 22  
Loan\_Amount\_Term 14  
Credit\_History 50  
Property\_Area 0  
Loan\_Status 0  
dtype: int64

Count of Approved and Rejected Applications[¶](#3znysh7)

In [10]:

df**.**groupby("Loan\_Status")["Loan\_Status"]**.**count()

Out[10]:

Loan\_Status  
N 192  
Y 422  
Name: Loan\_Status, dtype: int64

In [11]:

df**.**shape

Out[11]:

(614, 13)

In [12]:

df**.**info()

<class 'pandas.core.frame.DataFrame'>  
RangeIndex: 614 entries, 0 to 613  
Data columns (total 13 columns):  
 # Column Non-Null Count Dtype   
--- ------ -------------- -----   
 0 Loan\_ID 614 non-null object   
 1 Gender 601 non-null object   
 2 Married 611 non-null object   
 3 Dependents 599 non-null object   
 4 Education 614 non-null object   
 5 Self\_Employed 582 non-null object   
 6 ApplicantIncome 614 non-null int64   
 7 CoapplicantIncome 614 non-null float64  
 8 LoanAmount 592 non-null float64  
 9 Loan\_Amount\_Term 600 non-null float64  
 10 Credit\_History 564 non-null float64  
 11 Property\_Area 614 non-null object   
 12 Loan\_Status 614 non-null object   
dtypes: float64(4), int64(1), object(8)  
memory usage: 62.5+ KB

Data Transformation[¶](#2et92p0)

Missing values handling[¶](#tyjcwt)

In [13]:

df**.**isnull()**.**sum()

Out[13]:

Loan\_ID 0  
Gender 13  
Married 3  
Dependents 15  
Education 0  
Self\_Employed 32  
ApplicantIncome 0  
CoapplicantIncome 0  
LoanAmount 22  
Loan\_Amount\_Term 14  
Credit\_History 50  
Property\_Area 0  
Loan\_Status 0  
dtype: int64

In [14]:

df["Gender"]**.**fillna(df["Gender"]**.**mode()[0],inplace**=True**)  
df["Married"]**.**fillna(df["Married"]**.**mode()[0],inplace**=True**)  
df["Dependents"]**.**fillna(df["Dependents"]**.**mode()[0],inplace**=True**)  
df["Self\_Employed"]**.**fillna(df["Self\_Employed"]**.**mode()[0],inplace**=True**)  
df["LoanAmount"]**.**fillna(round(df["LoanAmount"]**.**mean(),1),inplace**=True**)  
df["Loan\_Amount\_Term"]**.**fillna(round(df["Loan\_Amount\_Term"]**.**mean(),1),inplace**=True**)  
df["Credit\_History"]**.**fillna(round(df["Credit\_History"]**.**mean(),1),inplace**=True**)

In [15]:

df**.**isnull()**.**sum()

Out[15]:

Loan\_ID 0  
Gender 0  
Married 0  
Dependents 0  
Education 0  
Self\_Employed 0  
ApplicantIncome 0  
CoapplicantIncome 0  
LoanAmount 0  
Loan\_Amount\_Term 0  
Credit\_History 0  
Property\_Area 0  
Loan\_Status 0  
dtype: int64

duplicate removal[¶](#3dy6vkm)

In [16]:

df**.**drop\_duplicates(inplace**=True**)

In [17]:

df**=**df**.**drop(columns**=**["Loan\_ID"])

In [18]:

df**.**head(3)

Out[18]:

|  | **Gender** | **Married** | **Dependents** | **Education** | **Self\_Employed** | **ApplicantIncome** | **CoapplicantIncome** | **LoanAmount** | **Loan\_Amount\_Term** | **Credit\_History** | **Property\_Area** | **Loan\_Status** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | Male | No | 0 | Graduate | No | 5849 | 0.0 | 146.4 | 360.0 | 1.0 | Urban | Y |
| **1** | Male | Yes | 1 | Graduate | No | 4583 | 1508.0 | 128.0 | 360.0 | 1.0 | Rural | N |
| **2** | Male | Yes | 0 | Graduate | Yes | 3000 | 0.0 | 66.0 | 360.0 | 1.0 | Urban | Y |

In [19]:

df**.**shape

Out[19]:

(614, 12)

Label Encoding ( change text values into Numeric)[¶](#1t3h5sf)

In [20]:

df**.**columns

Out[20]:

Index(['Gender', 'Married', 'Dependents', 'Education', 'Self\_Employed',  
 'ApplicantIncome', 'CoapplicantIncome', 'LoanAmount',  
 'Loan\_Amount\_Term', 'Credit\_History', 'Property\_Area', 'Loan\_Status'],  
 dtype='object')

In [21]:

col**=**['Gender', 'Married', 'Dependents', 'Education','Self\_Employed','Property\_Area', 'Loan\_Status']

In [22]:

col

Out[22]:

['Gender',  
 'Married',  
 'Dependents',  
 'Education',  
 'Self\_Employed',  
 'Property\_Area',  
 'Loan\_Status']

In [23]:

**from** sklearn.preprocessing **import** LabelEncoder

In [24]:

encoding**=**LabelEncoder()

In [25]:

**for** each **in** col:  
 df[each]**=**encoding**.**fit\_transform(df[each])

In [26]:

df**.**head(4)

Out[26]:

|  | **Gender** | **Married** | **Dependents** | **Education** | **Self\_Employed** | **ApplicantIncome** | **CoapplicantIncome** | **LoanAmount** | **Loan\_Amount\_Term** | **Credit\_History** | **Property\_Area** | **Loan\_Status** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 1 | 0 | 0 | 0 | 0 | 5849 | 0.0 | 146.4 | 360.0 | 1.0 | 2 | 1 |
| **1** | 1 | 1 | 1 | 0 | 0 | 4583 | 1508.0 | 128.0 | 360.0 | 1.0 | 0 | 0 |
| **2** | 1 | 1 | 0 | 0 | 1 | 3000 | 0.0 | 66.0 | 360.0 | 1.0 | 2 | 1 |
| **3** | 1 | 1 | 0 | 1 | 0 | 2583 | 2358.0 | 120.0 | 360.0 | 1.0 | 2 | 1 |

Data Correlation[¶](#4d34og8)

In [27]:

df**.**corr()

Out[27]:

|  | **Gender** | **Married** | **Dependents** | **Education** | **Self\_Employed** | **ApplicantIncome** | **CoapplicantIncome** | **LoanAmount** | **Loan\_Amount\_Term** | **Credit\_History** | **Property\_Area** | **Loan\_Status** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Gender** | 1.000000 | 0.364569 | 0.172914 | 0.045364 | -0.000525 | 0.058809 | 0.082912 | 0.107929 | -0.073567 | 0.013769 | -0.025752 | 0.017987 |
| **Married** | 0.364569 | 1.000000 | 0.334216 | 0.012304 | 0.004489 | 0.051708 | 0.075948 | 0.147141 | -0.100863 | 0.004467 | 0.004257 | 0.091478 |
| **Dependents** | 0.172914 | 0.334216 | 1.000000 | 0.055752 | 0.056798 | 0.118202 | 0.030430 | 0.163106 | -0.101054 | -0.036550 | -0.000244 | 0.010118 |
| **Education** | 0.045364 | 0.012304 | 0.055752 | 1.000000 | -0.010383 | -0.140760 | -0.062290 | -0.166999 | -0.077242 | -0.078887 | -0.065243 | -0.085884 |
| **Self\_Employed** | -0.000525 | 0.004489 | 0.056798 | -0.010383 | 1.000000 | 0.127180 | -0.016100 | 0.115260 | -0.033943 | -0.002445 | -0.030860 | -0.003700 |
| **ApplicantIncome** | 0.058809 | 0.051708 | 0.118202 | -0.140760 | 0.127180 | 1.000000 | -0.116605 | 0.565620 | -0.045242 | -0.013325 | -0.009500 | -0.004710 |
| **CoapplicantIncome** | 0.082912 | 0.075948 | 0.030430 | -0.062290 | -0.016100 | -0.116605 | 1.000000 | 0.187829 | -0.059675 | -0.005107 | 0.010522 | -0.059187 |
| **LoanAmount** | 0.107929 | 0.147141 | 0.163106 | -0.166999 | 0.115260 | 0.565620 | 0.187829 | 1.000000 | 0.038802 | -0.009416 | -0.044777 | -0.036414 |
| **Loan\_Amount\_Term** | -0.073567 | -0.100863 | -0.101054 | -0.077242 | -0.033943 | -0.045242 | -0.059675 | 0.038802 | 1.000000 | 0.001651 | -0.077620 | -0.020974 |
| **Credit\_History** | 0.013769 | 0.004467 | -0.036550 | -0.078887 | -0.002445 | -0.013325 | -0.005107 | -0.009416 | 0.001651 | 1.000000 | -0.002910 | 0.539071 |
| **Property\_Area** | -0.025752 | 0.004257 | -0.000244 | -0.065243 | -0.030860 | -0.009500 | 0.010522 | -0.044777 | -0.077620 | -0.002910 | 1.000000 | 0.032112 |
| **Loan\_Status** | 0.017987 | 0.091478 | 0.010118 | -0.085884 | -0.003700 | -0.004710 | -0.059187 | -0.036414 | -0.020974 | 0.539071 | 0.032112 | 1.000000 |

In [28]:

**import** seaborn **as** sns

In [33]:

plt**.**figure(figsize**=**(15,10))  
sns**.**heatmap(df**.**corr(),annot**=True**)  
plt**.**show()

Credit History has strong relation with target variable... strong relation b/w independent variables(Loan Amount~ApplicantIncome and Gender ~Married)can cuase multicollisionarity. so we need to drop one of these( which has week relation with target variable i-e applicantIncome)[¶](#2s8eyo1)

Input/out columns definition[¶](#17dp8vu)

In [34]:

x**=**df**.**iloc[:,:**-**1]**.**values  
y**=**df**.**iloc[:,**-**1]**.**values

In [35]:

x

Out[35]:

array([[ 1., 0., 0., ..., 360., 1., 2.],  
 [ 1., 1., 1., ..., 360., 1., 0.],  
 [ 1., 1., 0., ..., 360., 1., 2.],  
 ...,  
 [ 1., 1., 1., ..., 360., 1., 2.],  
 [ 1., 1., 2., ..., 360., 1., 2.],  
 [ 0., 0., 0., ..., 360., 0., 1.]])

In [36]:

y

Out[36]:

array([1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 0, 0, 1, 0, 1,  
 0, 0, 0, 1, 1, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1,  
 1, 1, 1, 1, 0, 1, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1, 1, 1, 0, 0, 0, 0,  
 0, 1, 1, 0, 1, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1,  
 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1,  
 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1,  
 1, 1, 1, 0, 0, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 0,  
 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 1, 0, 1,  
 1, 0, 1, 0, 0, 0, 1, 0, 1, 1, 0, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1,  
 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 0, 1, 0, 1,  
 0, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0,  
 1, 1, 1, 0, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1,  
 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1,  
 0, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 0, 1, 0,  
 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1,  
 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0,  
 1, 0, 1, 1, 1, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0, 0, 1, 0, 1, 1, 1, 0,  
 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1,  
 0, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0,  
 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1,  
 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 0, 1, 0, 1, 1,  
 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1,  
 1, 1, 0, 0, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1,  
 1, 0, 1, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 1, 0, 1, 1, 1,  
 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 0, 1, 0, 1,  
 0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 0, 1, 0,  
 1, 0, 0, 1, 0, 1, 1, 1, 1, 0, 1, 0, 0, 0, 1, 1, 1, 0, 1, 0, 1, 1,  
 1, 1, 0, 0, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0])

Normalization[¶](#3rdcrjn)

In [37]:

**from** sklearn.preprocessing **import** MinMaxScaler

In [38]:

scaling**=**MinMaxScaler()

In [39]:

x\_scaled**=**scaling**.**fit\_transform(x)

In [40]:

x\_scaled

Out[40]:

array([[1. , 0. , 0. , ..., 0.74358974, 1. ,  
 1. ],  
 [1. , 1. , 0.33333333, ..., 0.74358974, 1. ,  
 0. ],  
 [1. , 1. , 0. , ..., 0.74358974, 1. ,  
 1. ],  
 ...,  
 [1. , 1. , 0.33333333, ..., 0.74358974, 1. ,  
 1. ],  
 [1. , 1. , 0.66666667, ..., 0.74358974, 1. ,  
 1. ],  
 [0. , 0. , 0. , ..., 0.74358974, 0. ,  
 0.5 ]])

In [41]:

df**.**shape

Out[41]:

(614, 12)

Spliting data into training/testing set[¶](#26in1rg)

In [42]:

**from** sklearn.model\_selection **import** train\_test\_split

In [43]:

xtrain,xtest,ytrain,ytest**=**train\_test\_split(x\_scaled,y,test\_size**=**0.2,random\_state**=**123)

In [44]:

xtrain**.**shape

Out[44]:

(491, 11)

In [45]:

xtest**.**shape

Out[45]:

(123, 11)

ANN Model Creation for binary class result[¶](#lnxbz9)

In [46]:

**from** tensorflow.keras.models **import** Sequential  
**from** tensorflow.keras.layers **import** Dense

In [47]:

model**=**Sequential()  
model**.**add(Dense(30,activation**=**"relu"))  
model**.**add(Dense(20,activation**=**"relu"))  
model**.**add(Dense(1,activation**=**'sigmoid'))

In [48]:

model**.**compile(optimizer**=**"adam",loss**=**"binary\_crossentropy")

In [49]:

model**.**fit(xtrain,ytrain,epochs**=**50,batch\_size**=**1,validation\_data**=**(xtest,ytest))

Epoch 1/50  
 1/491 [..............................] - ETA: 0s - loss: 0.5564WARNING:tensorflow:Callbacks method `on\_train\_batch\_end` is slow compared to the batch time (batch time: 0.0000s vs `on\_train\_batch\_end` time: 0.0154s). Check your callbacks.  
491/491 [==============================] - 2s 4ms/step - loss: 0.5885 - val\_loss: 0.5701  
Epoch 2/50  
491/491 [==============================] - 1s 3ms/step - loss: 0.5228 - val\_loss: 0.5176  
Epoch 3/50  
491/491 [==============================] - 2s 3ms/step - loss: 0.4925 - val\_loss: 0.5026  
Epoch 4/50  
491/491 [==============================] - 2s 3ms/step - loss: 0.4807 - val\_loss: 0.5038  
Epoch 5/50  
491/491 [==============================] - 2s 3ms/step - loss: 0.4738 - val\_loss: 0.5027  
Epoch 6/50  
491/491 [==============================] - 2s 3ms/step - loss: 0.4679 - val\_loss: 0.5018  
Epoch 7/50  
491/491 [==============================] - 2s 3ms/step - loss: 0.4668 - val\_loss: 0.4977  
Epoch 8/50  
491/491 [==============================] - 2s 3ms/step - loss: 0.4648 - val\_loss: 0.5012  
Epoch 9/50  
491/491 [==============================] - 2s 3ms/step - loss: 0.4618 - val\_loss: 0.5149  
Epoch 10/50  
491/491 [==============================] - 2s 3ms/step - loss: 0.4624 - val\_loss: 0.5002  
Epoch 11/50  
491/491 [==============================] - 1s 3ms/step - loss: 0.4610 - val\_loss: 0.5026  
Epoch 12/50  
491/491 [==============================] - 2s 3ms/step - loss: 0.4591 - val\_loss: 0.5012  
Epoch 13/50  
491/491 [==============================] - 1s 3ms/step - loss: 0.4594 - val\_loss: 0.5060  
Epoch 14/50  
491/491 [==============================] - 2s 3ms/step - loss: 0.4563 - val\_loss: 0.4995  
Epoch 15/50  
491/491 [==============================] - 1s 2ms/step - loss: 0.4570 - val\_loss: 0.4980  
Epoch 16/50  
491/491 [==============================] - 1s 2ms/step - loss: 0.4543 - val\_loss: 0.5012  
Epoch 17/50  
491/491 [==============================] - 1s 2ms/step - loss: 0.4524 - val\_loss: 0.4991  
Epoch 18/50  
491/491 [==============================] - 1s 3ms/step - loss: 0.4507 - val\_loss: 0.5064  
Epoch 19/50  
491/491 [==============================] - 1s 2ms/step - loss: 0.4493 - val\_loss: 0.5087  
Epoch 20/50  
491/491 [==============================] - 2s 3ms/step - loss: 0.4475 - val\_loss: 0.5021  
Epoch 21/50  
491/491 [==============================] - 1s 2ms/step - loss: 0.4490 - val\_loss: 0.5024  
Epoch 22/50  
491/491 [==============================] - 1s 3ms/step - loss: 0.4459 - val\_loss: 0.5029  
Epoch 23/50  
491/491 [==============================] - 1s 1ms/step - loss: 0.4460 - val\_loss: 0.5072  
Epoch 24/50  
491/491 [==============================] - 1s 2ms/step - loss: 0.4451 - val\_loss: 0.4992  
Epoch 25/50  
491/491 [==============================] - 2s 3ms/step - loss: 0.4385 - val\_loss: 0.5133  
Epoch 26/50  
491/491 [==============================] - 2s 3ms/step - loss: 0.4413 - val\_loss: 0.5131  
Epoch 27/50  
491/491 [==============================] - 2s 3ms/step - loss: 0.4387 - val\_loss: 0.5041  
Epoch 28/50  
491/491 [==============================] - 1s 2ms/step - loss: 0.4409 - val\_loss: 0.5004  
Epoch 29/50  
491/491 [==============================] - 2s 3ms/step - loss: 0.4393 - val\_loss: 0.5088  
Epoch 30/50  
491/491 [==============================] - 1s 3ms/step - loss: 0.4375 - val\_loss: 0.5203  
Epoch 31/50  
491/491 [==============================] - 1s 3ms/step - loss: 0.4331 - val\_loss: 0.5009  
Epoch 32/50  
491/491 [==============================] - 1s 2ms/step - loss: 0.4323 - val\_loss: 0.5131  
Epoch 33/50  
491/491 [==============================] - 1s 2ms/step - loss: 0.4363 - val\_loss: 0.5075  
Epoch 34/50  
491/491 [==============================] - 1s 3ms/step - loss: 0.4295 - val\_loss: 0.5092  
Epoch 35/50  
491/491 [==============================] - 1s 2ms/step - loss: 0.4310 - val\_loss: 0.5050  
Epoch 36/50  
491/491 [==============================] - 2s 4ms/step - loss: 0.4328 - val\_loss: 0.5030  
Epoch 37/50  
491/491 [==============================] - 1s 3ms/step - loss: 0.4319 - val\_loss: 0.5103  
Epoch 38/50  
491/491 [==============================] - 1s 2ms/step - loss: 0.4250 - val\_loss: 0.5093  
Epoch 39/50  
491/491 [==============================] - 1s 2ms/step - loss: 0.4275 - val\_loss: 0.5055  
Epoch 40/50  
491/491 [==============================] - 1s 2ms/step - loss: 0.4205 - val\_loss: 0.5142  
Epoch 41/50  
491/491 [==============================] - 1s 3ms/step - loss: 0.4252 - val\_loss: 0.5039  
Epoch 42/50  
491/491 [==============================] - 1s 2ms/step - loss: 0.4223 - val\_loss: 0.5104  
Epoch 43/50  
491/491 [==============================] - 1s 3ms/step - loss: 0.4211 - val\_loss: 0.5110  
Epoch 44/50  
491/491 [==============================] - 2s 3ms/step - loss: 0.4198 - val\_loss: 0.5153  
Epoch 45/50  
491/491 [==============================] - 2s 4ms/step - loss: 0.4182 - val\_loss: 0.5185  
Epoch 46/50  
491/491 [==============================] - 1s 2ms/step - loss: 0.4215 - val\_loss: 0.5081  
Epoch 47/50  
491/491 [==============================] - 1s 3ms/step - loss: 0.4167 - val\_loss: 0.5386  
Epoch 48/50  
491/491 [==============================] - 2s 3ms/step - loss: 0.4101 - val\_loss: 0.5271  
Epoch 49/50  
491/491 [==============================] - 1s 2ms/step - loss: 0.4149 - val\_loss: 0.5242  
Epoch 50/50  
491/491 [==============================] - 1s 2ms/step - loss: 0.4150 - val\_loss: 0.5186

Out[49]:

<tensorflow.python.keras.callbacks.History at 0x1b1653438e0>

In [51]:

model**.**summary()

Model: "sequential"  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
Layer (type) Output Shape Param #   
=================================================================  
dense (Dense) (1, 30) 360   
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
dense\_1 (Dense) (1, 20) 620   
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
dense\_2 (Dense) (1, 1) 21   
=================================================================  
Total params: 1,001  
Trainable params: 1,001  
Non-trainable params: 0  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

In [54]:

los**=**pd**.**DataFrame(model**.**history**.**history)

In [55]:

plt**.**plot(los,marker**=**"o")  
plt**.**show()

In [58]:

**from** sklearn.metrics **import** confusion\_matrix,accuracy\_score,classification\_report

In [69]:

y\_predic**=**(model**.**predict(xtest)**>**0.5)**.**astype("int32")

In [72]:

confusion\_matrix(ytest,y\_predic)

Out[72]:

array([[18, 24],  
 [ 3, 78]], dtype=int64)

In [74]:

accuracy\_score(ytest,y\_predic)

Out[74]:

0.7804878048780488

In [76]:

print(classification\_report(ytest,y\_predic))

precision recall f1-score support  
  
 0 0.86 0.43 0.57 42  
 1 0.76 0.96 0.85 81  
  
 accuracy 0.78 123  
 macro avg 0.81 0.70 0.71 123  
weighted avg 0.80 0.78 0.76 123

In [ ]: