```
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import Sequential
from tensorflow.keras.layers import Dense
```

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np
import warnings
warnings.filterwarnings('ignore')
```

```
df = pd.read_csv('loan1.csv')
df.head()
```

	ID	Loan Amount	Funded Amount	Funded Amount Investor	Term	Batch Enrolled	Interest Rate	Grade	Sub Grade
0	65087372	10000	32236	12329.36286	59	BAT2522922	11.135007	B	C4
1	1450153	3609	11940	12191.99692	59	BAT1586599	12.237563	C	D3
2	1969101	28276	9311	21603.22455	59	BAT2136391	12.545884	F	D4
3	6651430	11170	6954	17877.15585	59	BAT2428731	16.731201	C	C3
4	14354669	16890	13226	13539.92667	59	BAT5341619	15.008300	C	D4

5 rows × 35 columns

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 12260 entries, 0 to 12259
Data columns (total 35 columns):
 #   Column                        Non-Null Count  Dtype  
---  ------                        --------------  -----  
 0   ID                            12260 non-null  int64  
 1   Loan Amount                   12260 non-null  int64  
 2   Funded Amount                 12260 non-null  int64  
 3   Funded Amount Investor        12260 non-null  float64
 4   Term                          12260 non-null  int64  
 5   Batch Enrolled                12260 non-null  object 
 6   Interest Rate                 12260 non-null  float64
 7   Grade                         12260 non-null  object 
 8   Sub Grade                     12260 non-null  object 
 9   Employment Duration           12260 non-null  object 
 10  Home Ownership                12260 non-null  float64
 11  Verification Status           12260 non-null  object 
 12  Payment Plan                  12260 non-null  object 
 13  Loan Title                    12260 non-null  object 
 14  Debit to Income               12260 non-null  float64
 15  Delinquency - two years       12260 non-null  int64  
 16  Inquires - six months         12260 non-null  int64  
 17  Open Account                  12260 non-null  int64  
 18  Public Record                 12260 non-null  int64  
 19  Revolving Balance             12260 non-null  int64  
 20  Revolving Utilities           12260 non-null  float64
 21  Total Accounts                12260 non-null  int64  
 22  Initial List Status           12260 non-null  object 
 23  Total Received Interest       12260 non-null  float64
 24  Total Received Late Fee       12260 non-null  float64
 25  Recoveries                    12260 non-null  float64
 26  Collection Recovery Fee       12260 non-null  float64
 27  Collection 12 months Medical  12259 non-null  float64
 28  Application Type              12259 non-null  object 
 29  Last week Pay                 12259 non-null  float64
 30  Accounts Delinquent           12259 non-null  float64
 31  Total Collection Amount       12259 non-null  float64
 32  Total Current Balance         12259 non-null  float64
 33  Total Revolving Credit Limit  12259 non-null  float64
 34  Loan Status                   12259 non-null  float64
dtypes: float64(16), int64(10), object(9)
memory usage: 3.3+ MB
```

```
df.describe()
```

	ID	Loan Amount	Funded Amount	Funded Amount Investor	Term	Int
count	1.226000e+04	12260.000000	12260.000000	12260.000000	12260.000000	12260.0
mean	2.539800e+07	16766.225204	15831.505628	14619.642177	58.170718	11.8
std	2.088998e+07	8379.000447	8191.734681	6805.325460	3.316191	3.7
min	1.299125e+06	1020.000000	1098.000000	1127.754818	36.000000	5.3
25%	6.559144e+06	9929.000000	9213.750000	9868.433307	58.000000	9.3
50%	1.774408e+07	15960.500000	13075.000000	12768.923180	59.000000	11.3
75%	4.220348e+07	21980.000000	21880.000000	18007.143105	59.000000	14.1
max	7.210185e+07	34986.000000	34999.000000	34987.513000	59.000000	27.0

8 rows × 26 columns

```
df.isnull().sum()
```

```
ID                              0
Loan Amount                     0
Funded Amount                   0
Funded Amount Investor          0
Term                            0
Batch Enrolled                  0
Interest Rate                   0
Grade                           0
Sub Grade                       0
Employment Duration             0
Home Ownership                  0
Verification Status             0
Payment Plan                    0
Loan Title                      0
Debit to Income                 0
Delinquency - two years         0
Inquires - six months           0
Open Account                    0
Public Record                   0
Revolving Balance               0
Revolving Utilities             0
Total Accounts                  0
Initial List Status             0
Total Received Interest         0
Total Received Late Fee         0
Recoveries                      0
Collection Recovery Fee         0
Collection 12 months Medical    1
Application Type                1
Last week Pay                   1
Accounts Delinquent             1
Total Collection Amount         1
Total Current Balance           1
Total Revolving Credit Limit    1
Loan Status                     1
dtype: int64
```

```
df
```

	ID	Loan Amount	Funded Amount	Funded Amount Investor	Term	Batch Enrolled	Interest Rate	Grade
0	65087372	10000	32236	12329.362860	59	BAT2522922	11.135007	B
1	1450153	3609	11940	12191.996920	59	BAT1586599	12.237563	C
2	1969101	28276	9311	21603.224550	59	BAT2136391	12.545884	F
3	6651430	11170	6954	17877.155850	59	BAT2428731	16.731201	C
4	14354669	16890	13226	13539.926670	59	BAT5341619	15.008300	C
...	...	...	...	...	...	...	...	...
12255	32295742	15455	9314	7522.923594	58	BAT2078974	11.135354	A
12256	2320565	21435	24905	8277.966325	36	BAT5525466	13.510502	F
12257	5029510	31578	12702	26495.936690	58	BAT5525466	8.650557	E
12258	8582824	16179	22659	15732.358370	58	BAT1586599	6.686504	F
12259	41128187	4477	7174	12923.034090	59	BAT4694572	9.954777	D

12260 rows × 35 columns

```
cat_df = df.select_dtypes(['object'])

num_df = df.select_dtypes(['int64','float64'])


from sklearn.preprocessing import LabelEncoder

le = LabelEncoder

for i in cat_df:
  le = LabelEncoder()
  cat_df[i]=le.fit_transform(cat_df[i])

cat_df
```

	Batch Enrolled	Grade	Sub Grade	Employment Duration	Verification Status	Payment Plan	Loan Title	Initial List Status
0	16	1	13	0	0	0	41	1
1	4	2	17	2	1	0	48	0
2	11	5	18	0	1	0	41	1
3	15	2	12	0	1	0	48	1
4	32	2	18	0	1	0	37	1
...	...	...	...	...	...	...	...	...
12255	10	0	14	2	0	0	48	0
12256	34	5	6	2	1	0	37	0
12257	34	4	11	0	1	0	37	0
12258	4	5	5	1	0	0	48	0
12259	29	3	7	0	1	0	37	1

12260 rows × 9 columns

```
df2 = pd.concat([num_df,cat_df],axis=1)
df2
```

o e	Delinquency - two years	Inquires - six months	...	Loan Status	Batch Enrolled	Grade	Sub Grade	Employment Duration	Verificati Sta
3	1	0	...	0.0	16	1	13	0	
9	0	0	...	0.0	4	2	17	2	
9	0	0	...	0.0	11	5	18	0	
0	1	0	...	0.0	15	2	12	0	
6	1	3	...	0.0	32	2	18	0	
.	...	...	...	...	...	...	...	...	
3	0	0	...	0.0	10	0	14	2	
5	0	0	...	0.0	34	5	6	2	
7	0	0	...	0.0	34	4	11	0	
2	0	0	...	0.0	4	5	5	1	
0	0	0	...	NaN	29	3	7	0	

```
x = df2.drop(['Loan Status'],axis=1).values
x
```

```
array([[6.5087372e+07, 1.0000000e+04, 3.2236000e+04, ..., 4.1000000e+01,
        1.0000000e+00, 0.0000000e+00],
       [1.4501530e+06, 3.6090000e+03, 1.1940000e+04, ..., 4.8000000e+01,
        0.0000000e+00, 0.0000000e+00],
       [1.9691010e+06, 2.8276000e+04, 9.3110000e+03, ..., 4.1000000e+01,
        1.0000000e+00, 0.0000000e+00],
       ...,
       [5.0295100e+06, 3.1578000e+04, 1.2702000e+04, ..., 3.7000000e+01,
        0.0000000e+00, 0.0000000e+00],
       [8.5828240e+06, 1.6179000e+04, 2.2659000e+04, ..., 4.8000000e+01,
        0.0000000e+00, 0.0000000e+00],
       [4.1128187e+07, 4.4770000e+03, 7.1740000e+03, ..., 3.7000000e+01,
        1.0000000e+00, 2.0000000e+00]])
```

```
y = df2['Loan Status'].values
y
```

```
array([ 0.,  0.,  0., ...,  0.,  0., nan])
```

```
from sklearn.preprocessing import StandardScaler

sc = StandardScaler()

x = sc.fit_transform(x)
```

```
from sklearn.model_selection import train_test_split

xtrain,xtest,ytrain,ytest = train_test_split(x,y,test_size=0.3,random_state=1)
```

```
from scipy.stats.morestats import optimize
```

```
ann =Sequential()
```

```
ann.add(Dense(units=5,activation="relu"))
ann.add(Dense(units=1,activation='sigmoid'))
```

```
ann.compile(optimizer="adam",loss='binary_crossentropy',metrics=['accuracy'])
ann.fit(xtrain,ytrain,batch_size=30,epochs=100)

ypred = ann.predict(xtest)
```

```
     287/287 [==============================] - 0s 2ms/step - loss: nan - accuracy: 0.9041
     Epoch 73/100
     287/287 [==============================] - 0s 2ms/step - loss: nan - accuracy: 0.9041
     Epoch 74/100
     287/287 [==============================] - 0s 2ms/step - loss: nan - accuracy: 0.9041
     Epoch 75/100
     287/287 [==============================] - 0s 2ms/step - loss: nan - accuracy: 0.9041
     Epoch 76/100
     287/287 [==============================] - 0s 2ms/step - loss: nan - accuracy: 0.9041
     Epoch 77/100
     287/287 [==============================] - 0s 2ms/step - loss: nan - accuracy: 0.9041
     Epoch 78/100
     287/287 [==============================] - 1s 2ms/step - loss: nan - accuracy: 0.9041
     Epoch 79/100
     287/287 [==============================] - 1s 2ms/step - loss: nan - accuracy: 0.9041
     Epoch 80/100
     287/287 [==============================] - 1s 2ms/step - loss: nan - accuracy: 0.9041
     Epoch 81/100
     287/287 [==============================] - 1s 2ms/step - loss: nan - accuracy: 0.9041
     Epoch 82/100
     287/287 [==============================] - 0s 2ms/step - loss: nan - accuracy: 0.9041
     Epoch 83/100
     287/287 [==============================] - 1s 2ms/step - loss: nan - accuracy: 0.9041
     Epoch 84/100
     287/287 [==============================] - 0s 2ms/step - loss: nan - accuracy: 0.9041
     Epoch 85/100
     287/287 [==============================] - 1s 2ms/step - loss: nan - accuracy: 0.9041
     Epoch 86/100
     287/287 [==============================] - 1s 2ms/step - loss: nan - accuracy: 0.9041
     Epoch 87/100
     287/287 [==============================] - 1s 2ms/step - loss: nan - accuracy: 0.9041
     Epoch 88/100
     287/287 [==============================] - 1s 2ms/step - loss: nan - accuracy: 0.9041
     Epoch 89/100
     287/287 [==============================] - 0s 2ms/step - loss: nan - accuracy: 0.9041
     Epoch 90/100
     287/287 [==============================] - 1s 2ms/step - loss: nan - accuracy: 0.9041
     Epoch 91/100
     287/287 [==============================] - 1s 2ms/step - loss: nan - accuracy: 0.9041
     Epoch 92/100
     287/287 [==============================] - 1s 2ms/step - loss: nan - accuracy: 0.9041
     Epoch 93/100
     287/287 [==============================] - 0s 2ms/step - loss: nan - accuracy: 0.9041
     Epoch 94/100
     287/287 [==============================] - 0s 2ms/step - loss: nan - accuracy: 0.9041
     Epoch 95/100
     287/287 [==============================] - 0s 2ms/step - loss: nan - accuracy: 0.9041
     Epoch 96/100
     287/287 [==============================] - 0s 2ms/step - loss: nan - accuracy: 0.9041
     Epoch 97/100
     287/287 [==============================] - 1s 2ms/step - loss: nan - accuracy: 0.9041
     Epoch 98/100
     287/287 [==============================] - 1s 2ms/step - loss: nan - accuracy: 0.9041
     Epoch 99/100
     287/287 [==============================] - 0s 2ms/step - loss: nan - accuracy: 0.9041
     Epoch 100/100
     287/287 [==============================] - 0s 2ms/step - loss: nan - accuracy: 0.9041
     115/115 [==============================] - 0s 1ms/step
```

```
from sklearn.preprocessing import StandardScaler

sc = StandardScaler()

xtrain = sc.fit_transform(xtrain)

xtest = sc.fit_transform(xtest)
```

```
ann =Sequential()
```

```
ann.add(Dense(units=5,activation="relu"))
ann.add(Dense(units=1,activation='sigmoid'))


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

ann.fit(xtrain,ytrain,batch_size=30,epochs=100)

ypred = ann.predict(xtest)
```

```
287/287 [==============================] - 0s 2ms/step - loss: nan - accuracy: 0.9041
Epoch 73/100
287/287 [==============================] - 0s 2ms/step - loss: nan - accuracy: 0.9041
Epoch 74/100
287/287 [==============================] - 0s 2ms/step - loss: nan - accuracy: 0.9041
Epoch 75/100
287/287 [==============================] - 0s 2ms/step - loss: nan - accuracy: 0.9041
Epoch 76/100
287/287 [==============================] - 0s 2ms/step - loss: nan - accuracy: 0.9041
Epoch 77/100
287/287 [==============================] - 0s 2ms/step - loss: nan - accuracy: 0.9041
Epoch 78/100
287/287 [==============================] - 0s 2ms/step - loss: nan - accuracy: 0.9041
Epoch 79/100
287/287 [==============================] - 1s 2ms/step - loss: nan - accuracy: 0.9041
Epoch 80/100
287/287 [==============================] - 1s 2ms/step - loss: nan - accuracy: 0.9041
Epoch 81/100
287/287 [==============================] - 1s 2ms/step - loss: nan - accuracy: 0.9041
Epoch 82/100
287/287 [==============================] - 0s 2ms/step - loss: nan - accuracy: 0.9041
Epoch 83/100
287/287 [==============================] - 0s 2ms/step - loss: nan - accuracy: 0.9041
Epoch 84/100
287/287 [==============================] - 0s 2ms/step - loss: nan - accuracy: 0.9041
Epoch 85/100
287/287 [==============================] - 0s 2ms/step - loss: nan - accuracy: 0.9041
Epoch 86/100
287/287 [==============================] - 0s 2ms/step - loss: nan - accuracy: 0.9041
Epoch 87/100
287/287 [==============================] - 0s 2ms/step - loss: nan - accuracy: 0.9041
Epoch 88/100
287/287 [==============================] - 0s 2ms/step - loss: nan - accuracy: 0.9041
Epoch 89/100
287/287 [==============================] - 0s 2ms/step - loss: nan - accuracy: 0.9041
Epoch 90/100
287/287 [==============================] - 0s 2ms/step - loss: nan - accuracy: 0.9041
Epoch 91/100
287/287 [==============================] - 0s 2ms/step - loss: nan - accuracy: 0.9041
Epoch 92/100
287/287 [==============================] - 0s 2ms/step - loss: nan - accuracy: 0.9041
Epoch 93/100
287/287 [==============================] - 0s 2ms/step - loss: nan - accuracy: 0.9041
Epoch 94/100
287/287 [==============================] - 0s 2ms/step - loss: nan - accuracy: 0.9041
Epoch 95/100
287/287 [==============================] - 0s 2ms/step - loss: nan - accuracy: 0.9041
Epoch 96/100
287/287 [==============================] - 0s 2ms/step - loss: nan - accuracy: 0.9041
Epoch 97/100
287/287 [==============================] - 0s 2ms/step - loss: nan - accuracy: 0.9041
Epoch 98/100
287/287 [==============================] - 0s 2ms/step - loss: nan - accuracy: 0.9041
Epoch 99/100
287/287 [==============================] - 0s 2ms/step - loss: nan - accuracy: 0.9041
Epoch 100/100
287/287 [==============================] - 0s 2ms/step - loss: nan - accuracy: 0.9041
115/115 [==============================] - 0s 1ms/step
```

✓ 51s completed at 12:01 AM