S. B. JAIN INSTITUTE OF TECHNOLOGY, MANAGEMENT & RESEARCH, NAGPUR.

Practical No. 5

Aim: Design and implement Artificial Neural Network in Machine Learning.

Name of Student : **Shrutika Pradeep Bagdi**

Roll No. : **CS22130**

Semester/Year : **6^{th} / 3^{rd}**

Academic Session : **2024-2025**

Date of Performance :

Date of Submission :

Aim: Design and implement Artificial Neural Network in Machine Learning.

OBJECTIVE/EXPECTED LEARNING OUTCOME:

The objectives and expected learning outcome of this practical are:

It is designed to analyse and process information as humans. Artificial Neural Network has self-learning capabilities to produce better results as more data is available.

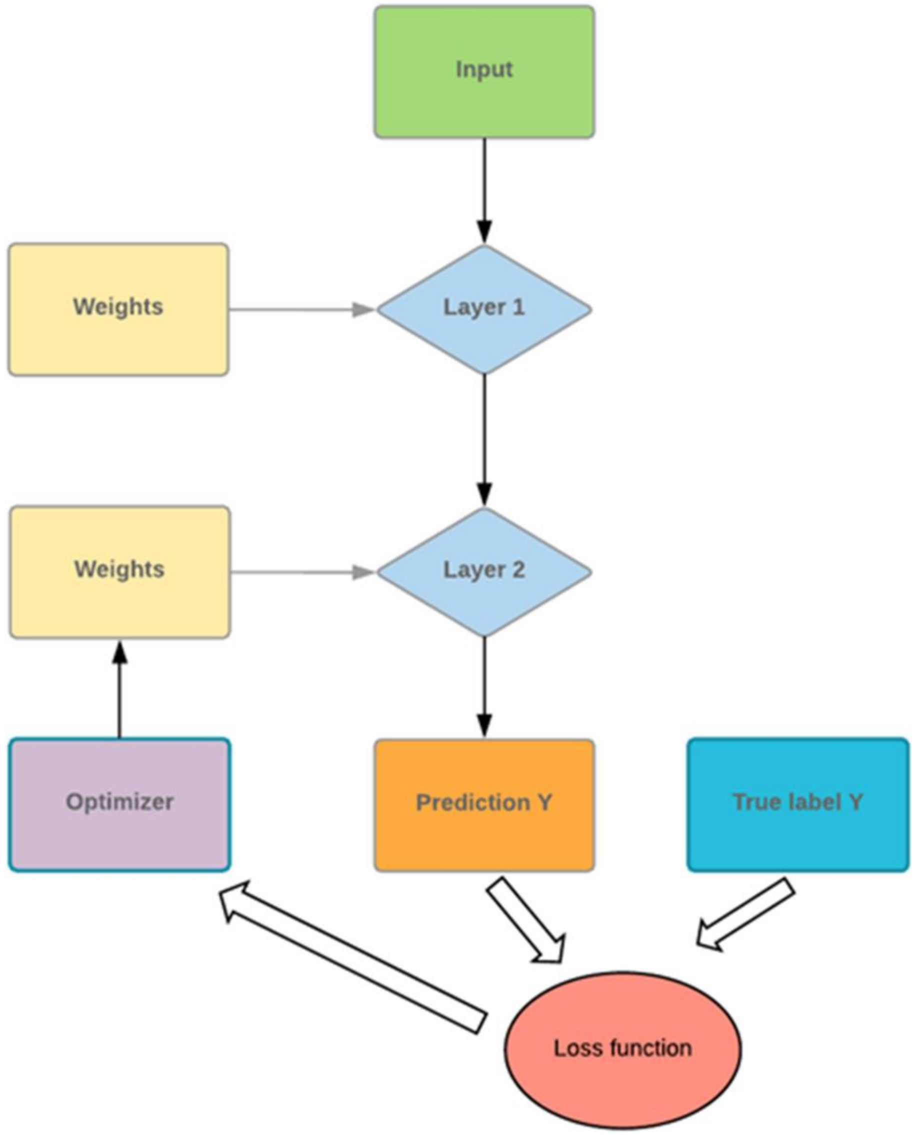
THEORY:

Neural networks are parallel computing devices, which is basically an attempt to make a computer model of the brain. The main objective is to develop a system to perform various computational tasks faster than the traditional systems. These tasks include pattern recognition and classification, approximation, optimization, and data clustering.

ANNs (Artificial Neural Network) is at the very core of Deep Learning an advanced version of Machine Learning techniques. ANNs are versatile, adaptive, and scalable, making them appropriate to tackle large datasets and highly complex Machine Learning tasks such as image classification (e.g., Google Images), speech recognition (e.g., Apple's Siri), video recommendation (e.g., YouTube), or analyzing sentiments among customers (e.g. Twitter Sentiment Analyzer).

What is an Artificial Neural Network?

An **Artificial Neural Network** (ANN) is a computer system inspired by biological neural networks for creating artificial brains based on the collection of connected units called artificial neurons. It is designed to analyse and process information as humans. Artificial Neural Network has self-learning capabilities to produce better results as more data is available.



Artificial Neural Network

An Artificial Neural Network (ANN) is composed of four principal objects:

- **Layers**: all the learning occurs in the layers. There are 3 layers 1) Input 2) Hidden and 3) Output
- **Feature and label**: Input data to the network (features) and output from the network (labels)
- **Loss function**: Metric used to estimate the performance of the learning phase
- **Optimizer**: Improve the learning by updating the knowledge in the network

A neural network will take the input data and push them into an ensemble of layers. The network needs to evaluate its performance with a loss function. The loss function gives to the network an idea of the path it needs to take before it masters the knowledge. The network needs to improve its knowledge with the help of an optimizer.

If you take a look at the figure above, you will understand the underlying mechanism.

The program takes some input values and pushes them into two fully connected layers. Imagine you have a math problem, the first thing you do is to read the corresponding chapter to solve the problem. You apply your new knowledge to solve the problem. There is a high chance you will not score very well. It is the same for a network. The first time it sees the data and makes a prediction, it will not match perfectly with the actual data.

To improve its knowledge, the network uses an optimizer. In our analogy, an optimizer can be thought of as rereading the chapter. You gain new insights/lesson by reading again. Similarly, the network uses the optimizer, updates its knowledge, and tests its new knowledge to check how much it still needs to learn. The program will repeat this step until it makes the lowest error possible.

In our math problem analogy, it means you read the textbook chapter many times until you thoroughly understand the course content. Even after reading multiple times, if you keep making an error, it means you reached the knowledge capacity with the current material. You need to use different textbook or test different method to improve your score. For a neural network, it is the same process. If the error is far from 100%, but the curve is flat, it means with the current architecture; it cannot learn anything else. The network has to be better optimized to improve the knowledge.

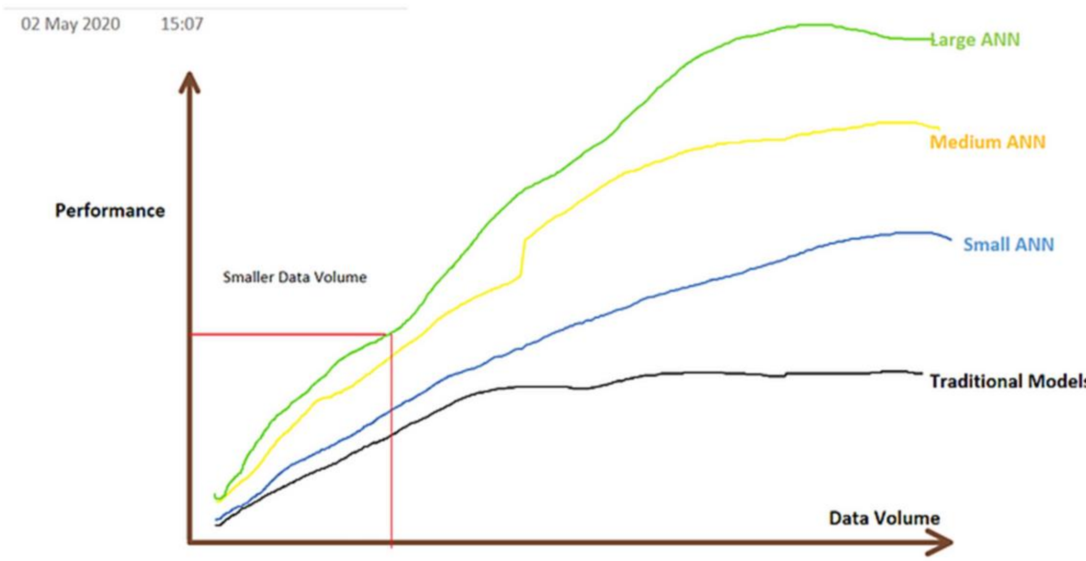


Figure 1. Illustrates the Variation of performance with an increase in data volume. Most algorithms produce similar performance measures for smaller datasets, however, Neural Networks works best once the data volume expands beyond a certain threshold. (image developed by the author using one note)

Program for reference:

```
X = data_frame.drop(columns='label', axis=1)
Y = data_frame['label']
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_state=2)
print(X.shape, X_train.shape, X_test.shape)
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()

X_train_std = scaler.fit_transform(X_train)
X_test_std = scaler.transform(X_test)
import tensorflow as tf
tf.random.set_seed(3)
from tensorflow import keras
model = keras.Sequential([
                    keras.layers.Flatten(input_shape=(30,)),
                    keras.layers.Dense(20, activation='relu'),
                    keras.layers.Dense(2, activation='sigmoid')
])
model.compile(optimizer='adam',
          loss='sparse_categorical_crossentropy',
          metrics=['accuracy'])
loss, accuracy = model.evaluate(X_test_std, Y_test)
print(accuracy)
Y_pred = model.predict(X_test_std)
```

PROGRAM CODE:

OUTPUT (SCREENSHOT):

```
#Shrutika Pradeep Bagdi (CS22130)
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
#Shrutika Pradeep Bagdi (CS22130)
import warnings
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

```
#Shrutika Pradeep Bagdi (CS22130)
path="/content/drive/MyDrive/Machine Learning (ML)/churn.csv"
df=pd.read_csv(path)
df.head()
```

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
0	1	15634602	Hargrave	619	France	Female	42	2	0.00	1	1	1	101348.88	1
1	2	15647311	Hill	608	Spain	Female	41	1	83807.86	1	0	1	112542.58	0
2	3	15619304	Onio	502	France	Female	42	8	159660.80	3	1	0	113931.57	1
3	4	15701354	Boni	699	France	Female	39	1	0.00	2	0	0	93826.63	0
4	5	15737888	Mitchell	850	Spain	Female	43	2	125510.82	1	1	1	79084.10	0

```
#Shrutika Pradeep Bagdi (CS22130)
df.shape
```

```
(10000, 14)
```

```
#Shrutika Pradeep Bagdi (CS22130)
df.columns
```

```
Index(['RowNumber', 'CustomerId', 'Surname', 'CreditScore', 'Geography',
       'Gender', 'Age', 'Tenure', 'Balance', 'NumOfProducts', 'HasCrCard',
       'IsActiveMember', 'EstimatedSalary', 'Exited'],
      dtype='object')
```

```
#Shrutika Pradeep Bagdi (CS22130)
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
```

```
#Shrutika Pradeep Bagdi (CS22130)
df.describe()
```

	RowNumber	CustomerId	CreditScore	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
count	10000.00000	1.000000e+04	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.00000	10000.000000	10000.000000	10000.000000
mean	5000.50000	1.569094e+07	650.528800	38.921800	5.012800	76485.889288	1.530200	0.70550	0.515100	100090.239881	0.203700
std	2886.89568	7.193619e+04	96.653299	10.487806	2.892174	62397.405202	0.581654	0.45584	0.499797	57510.492818	0.402769
min	1.00000	1.556570e+07	350.000000	18.000000	0.000000	0.000000	1.000000	0.00000	0.000000	11.580000	0.000000
25%	2500.75000	1.562853e+07	584.000000	32.000000	3.000000	0.000000	1.000000	0.00000	0.000000	51002.110000	0.000000
50%	5000.50000	1.569074e+07	652.000000	37.000000	5.000000	97198.540000	1.000000	1.00000	1.000000	100193.915000	0.000000
75%	7500.25000	1.575323e+07	718.000000	44.000000	7.000000	127644.240000	2.000000	1.00000	1.000000	149388.247500	0.000000
max	10000.00000	1.581569e+07	850.000000	92.000000	10.000000	250898.090000	4.000000	1.00000	1.000000	199992.480000	1.000000

```
#Shrutika Pradeep Bagdi (CS22130)
df.isnull().any().any()
```

```
False
```

```
#Shrutika Pradeep Bagdi (CS22130)
df.isnull().sum()
```

	0
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

+ Code + Text Connect

```
[ ] #Shrutika Pradeep Bagdi (CS22130)
    df[df.isnull().any(axis=1)]
```

RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited

```
[ ] #Shrutika Pradeep Bagdi (CS22130)
    X = df.iloc[:,3:-1]
```

```
[ ] #Shrutika Pradeep Bagdi (CS22130)
    X.shape
```

```
(10000, 10)
```

```
[ ] #Shrutika Pradeep Bagdi (CS22130)
    y=df.iloc[:,-1]
```

```
[ ] #Shrutika Pradeep Bagdi (CS22130)
    y.shape
```

```
(10000,)
```

```
[ ] #Shrutika Pradeep Bagdi (CS22130)
    X = pd.get_dummies(X, dtype = int)
```

+ Code + Text Connect

```
[ ] #Shrutika Pradeep Bagdi (CS22130)
    X.shape
```

```
(10000, 13)
```

```
#Shrutika Pradeep Bagdi (CS22130)
print (X)
```

```
      CreditScore  Age  Tenure    Balance  NumOfProducts  HasCrCard  \
0             619   42       2       0.00              1          1
1             608   41       1   83807.86              1          0
2             502   42       8  159660.80              3          1
3             699   39       1       0.00              2          0
4             850   43       2  125510.82              1          1
...           ...  ...     ...        ...            ...        ...
9995          771   39       5       0.00              2          1
9996          516   35      10   57369.61              1          1
9997          709   36       7       0.00              1          0
9998          772   42       3   75075.31              2          1
9999          792   28       4  130142.79              1          1

      IsActiveMember  EstimatedSalary  Geography_France  Geography_Germany  \
0                  1        101348.88                 1                  0
1                  1        112542.58                 0                  0
2                  0        113931.57                 1                  0
3                  0         93826.63                 1                  0
4                  1         79084.10                 0                  0
...              ...              ...               ...                ...
9995               0         96270.64                 1                  0
9996               1        101699.77                 1                  0
9997               1         42085.58                 1                  0
9998               0         92888.52                 0                  1
```

+ Code + Text Connect Gemini

```
[ ] #Shrutika Pradeep Bagdi (CS22130)
    X.columns
```

```
Index(['CreditScore', 'Age', 'Tenure', 'Balance', 'NumOfProducts', 'HasCrCard',
       'IsActiveMember', 'EstimatedSalary', 'Geography_France',
       'Geography_Germany', 'Geography_Spain', 'Gender_Female', 'Gender_Male'],
      dtype='object')
```

```
[ ] #Shrutika Pradeep Bagdi (CS22130)
    from sklearn.model_selection import train_test_split
    X_train, X_test, y_train, y_test = train_test_split(X, y, train_size = 0.7, test_size = 0.25, random_state = 42)
```

```
#Shrutika Pradeep Bagdi (CS22130)
X_train.shape,X_test.shape
```

```
((7000, 13), (2500, 13))
```

```
[ ] #Shrutika Pradeep Bagdi (CS22130)
    X_train.head(5)
```

	CreditScore	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Geography_France	Geography_Germany	Geography_Spain	Gender_Female	Gender_Male
4901	673	59	0	178058.06	2	0	1	21063.71	1	0	0	0	1
4375	850	41	8	60880.68	1	1	0	31825.84	0	1	0	0	1
6698	725	31	6	0.00	1	0	0	61326.43	1	0	0	1	0
9805	644	33	7	174571.36	1	0	1	43943.09	1	0	0	0	1

```python
#Shrutika Pradeep Bagdi (CS22130)
X_test.head(5)
```

	CreditScore	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Geography_France	Geography_Germany	Geography_Spain	Gender_Female	Gender_Male
6252	596	32	3	96709.07	2	0	0	41788.37	0	1	0	0	1
4684	623	43	1	0.00	2	1	1	146379.30	1	0	0	0	1
1731	601	44	4	0.00	2	1	0	58561.31	0	0	1	1	0
4742	506	59	8	119152.10	2	1	1	170679.74	0	1	0	0	1
4521	560	27	7	124995.98	1	1	1	114669.79	0	0	1	1	0

```python
#Shrutika Pradeep Bagdi (CS22130)
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

```python
#Shrutika Pradeep Bagdi (CS22130)
X_train
```

```
array([[ 0.22073665,  1.91104421, -1.73091421, ..., -0.5736112 ,
        -0.9147613 ,  0.9147613 ],
       [ 2.06144677,  0.2000991 ,  1.03807384, ..., -0.5736112 ,
        -0.9147613 ,  0.9147613 ],
       [ 0.76151024, -0.75042596,  0.34582683, ..., -0.5736112 ,
         1.09318135, -1.09318135],
       ...,
```

```python
#Shrutika Pradeep Bagdi (CS22130)
X_test
```

```
array([[-5.80024249e-01, -6.55373449e-01, -6.92543692e-01, ...,
        -5.73611200e-01, -9.14761305e-01,  9.14761305e-01],
       [-2.99237960e-01,  3.90204116e-01, -1.38479071e+00, ...,
        -5.73611200e-01, -9.14761305e-01,  9.14761305e-01],
       [-5.28026788e-01,  4.85256622e-01, -3.46420185e-01, ...,
         1.74334114e+00,  1.09318135e+00, -1.09318135e+00],
       ...,
       [ 1.20868841e+00, -1.32074099e+00, -1.73091421e+00, ...,
        -5.73611200e-01, -9.14761305e-01,  9.14761305e-01],
       [ 3.97528018e-01, -2.75163426e-01,  3.45826830e-01, ...,
        -5.73611200e-01,  1.09318135e+00, -1.09318135e+00],
       [-4.96828312e-01, -1.32074099e+00, -2.96677292e-04, ...,
        -5.73611200e-01,  1.09318135e+00, -1.09318135e+00]])
```

```python
#Shrutika Pradeep Bagdi (CS22130)
import tensorflow as tf
```

```python
#Shrutika Pradeep Bagdi (CS22130)
#Initialising
ann = tf.keras.models.Sequential()
```

```python
#Shrutika Pradeep Bagdi (CS22130)
#Creating Hidden Layer
#Adding First Hidden Layer
ann.add(tf.keras.layers.Dense(units=6, activation='relu'))
```

```python
#Shrutika Pradeep Bagdi (CS22130)
#Adding Second Hidden
ann.add(tf.keras.layers.Dense(units=6, activation='relu'))
```

```python
#Shrutika Pradeep Bagdi (CS22130)
#Adding Output Layer
ann.add(tf.keras.layers.Dense(units=1, activation='sigmoid'))
```

```python
#Shrutika Pradeep Bagdi (CS22130)
# Compiling the ANN
ann.compile(optimizer="adam",loss="binary_crossentropy",metrics=['accuracy'])
```

```python
#Shrutika Pradeep Bagdi (CS22130)
#Fitting the ANN to the traing set
ann.fit(X_train,y_train,batch_size=32,epochs=100)
```

```
219/219 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.8634 - loss: 0.3281
Epoch 73/100
219/219 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.8654 - loss: 0.3390
Epoch 74/100
219/219 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step - accuracy: 0.8673 - loss: 0.3287
Epoch 75/100
219/219 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - accuracy: 0.8649 - loss: 0.3305
Epoch 76/100
219/219 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.8664 - loss: 0.3355
Epoch 77/100
219/219 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.8644 - loss: 0.3376
Epoch 78/100
219/219 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - accuracy: 0.8599 - loss: 0.3360
Epoch 79/100
```

```
219/219 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.8624 - loss: 0.3407
Epoch 99/100
219/219 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.8735 - loss: 0.3153
Epoch 100/100
219/219 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.8655 - loss: 0.3268
<keras.src.callbacks.history.History at 0x78b64cadbf10>
```

```python
#Shrutika Pradeep Bagdi (CS22130)
#Predicting the Test set  result
y_pred = ann.predict(X_test)
y_pred = (y_pred > 0.5)
y_pred
```

```
79/79 ━━━━━━━━━━━━━━━━━━━━ 1s 7ms/step
array([[False],
       [False],
       [False],
       ...,
       [False],
       [False],
       [False]])
```

```python
#Shrutika Pradeep Bagdi (CS22130)
from sklearn.metrics import accuracy_score, classification_report,confusion_matrix
```

```python
#Shrutika Pradeep Bagdi (CS22130)
score = accuracy_score(y_pred, y_test)
print(score)
```

```
0.8624
```

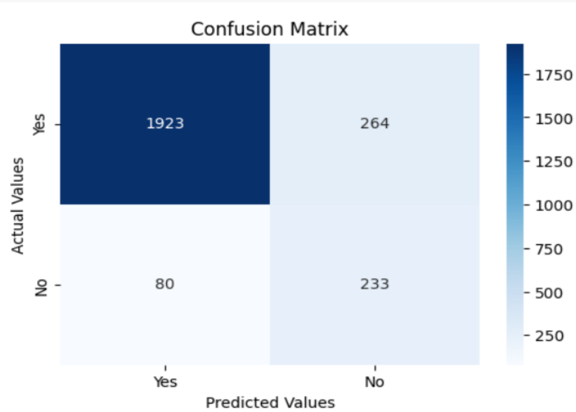
```python
#Shrutika Pradeep Bagdi (CS22130)
print(classification_report(y_pred, y_test))
cm = confusion_matrix(y_pred, y_test)
```

```
              precision    recall  f1-score   support

       False       0.96      0.88      0.92      2187
        True       0.47      0.74      0.58       313

    accuracy                           0.86      2500
   macro avg       0.71      0.81      0.75      2500
weighted avg       0.90      0.86      0.88      2500
```

```python
#Shrutika Pradeep Bagdi (CS22130)
print("Confusion Matrix:",confusion_matrix(y_pred, y_test))
```

```
Confusion Matrix: [[1923  264]
 [  80  233]]
```

```python
#Shrutika Pradeep Bagdi (CS22130)
import seaborn as sns
import matplotlib.pyplot as plt
plt.figure(figsize=(6, 4))
sns.heatmap(cm, annot=True, cmap='Blues', fmt='g',
            xticklabels=['Yes', 'No'], yticklabels=['Yes', 'No'])
plt.xlabel('Predicted Values')
plt.ylabel('Actual Values')
plt.title('Confusion Matrix')
plt.show()
```



CONCLUSION:

_ _

_ _

_ _

_ _

DISCUSSION AND VIVA VOCE:

Q1 What is Artificial Neural Network?

Q2: What do you mean by Perceptron?

Q3: What is the use of loss function?

Q4: What is the gradient of the binary step function?

Q5: Why are activation function introduced?

REFERENCE:

- https://en.wikipedia.org/wiki/Artificial_neural_network
- https://www.sciencedirect.com/topics/earth-and-planetary-sciences/artificial-neural-network