Lab Assignment No. 3

Write a Python Program using Perceptron Neural Network to recognise even and odd numbers. Given numbers are in ASCII form 0 to 9

Code:

```
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
import seaborn as sns
```

```
data = {"Numberes" : [1,3,5,4,2,9,7,6,8], "Tag" : [0,0,0, 1,1,0,0,1,1]}
df = pd.DataFrame(data)
```

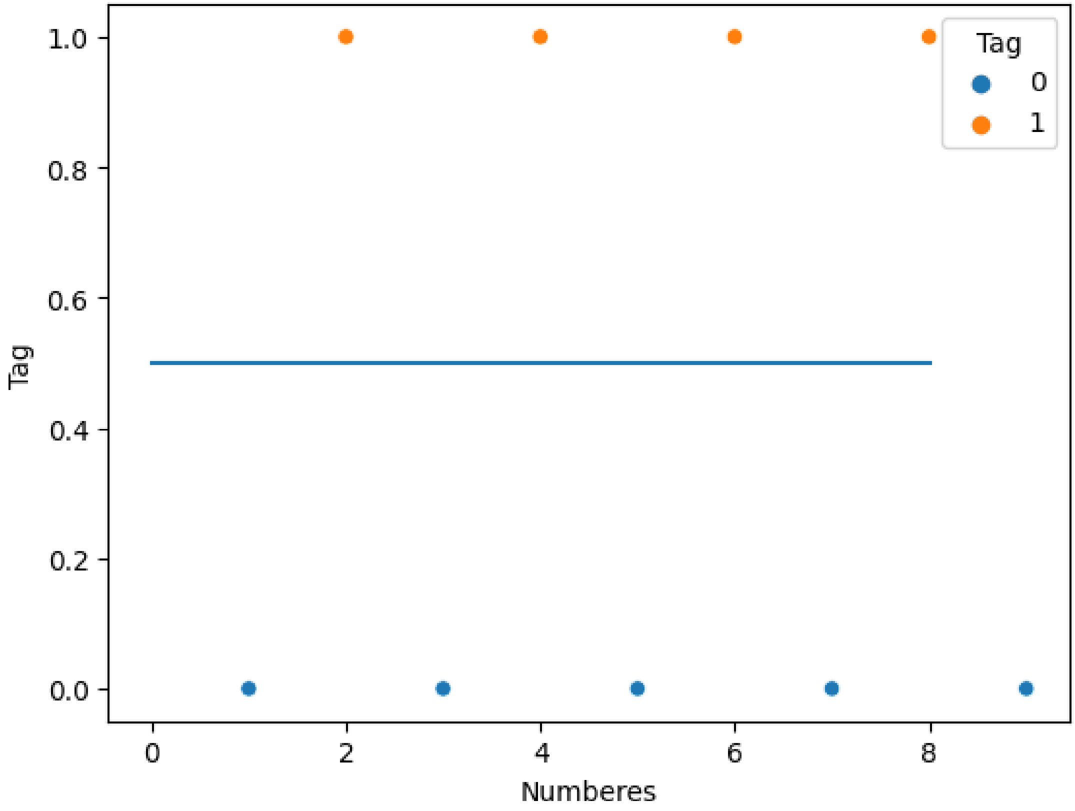
```
df
```

```
   Numberes  Tag
0         1    0
1         3    0
2         5    0
3         4    1
4         2    1
5         9    0
6         7    0
7         6    1
8         8    1
```

```
x = df["Numberes"]
y = df["Tag"]
```

```
sns.scatterplot(x=df["Numberes"],y=y, hue=y)
plt.plot([0.5 for _ in df["Tag"] ])
```

```
[<matplotlib.lines.Line2D at 0x7f3457dc1ac0>]
```



```
2 * np.random.random((10, 1)) - 1
```

```
array([[-0.35812149],
       [ 0.47412758],
       [ 0.62511668],
       [-0.89430268],
       [ 0.32154228],
       [ 0.08212657],
       [ 0.4911432 ],
       [ 0.7819753 ],
       [ 0.92660091],
       [ 0.62252329]])
```

```
array([[ 0.2082653 ],
       [-0.74252417],
       [-0.77242322],
       [-0.86954873],
       [ 0.33953798],
       [ 0.74940269],
       [-0.29060187],
       [-0.72214394],
       [-0.49388729],
       [-0.09347683]])
```

```
int(bin(2)[2:])
```

```
10
```

```
a = [np.random.choice([0,1]) for _ in range(4)]
a
```

```
[0, 1, 0, 1]
```

```
1 if 8>0 else 0
```

```
1
```

```
np.ones(4)
```

```
array([1., 1., 1., 1.])
```

```
a = 2 + np.dot([1,2, 4], [2, 2,2])
a
```

```
16
```

#Class for binary input

```
class Perceptron():
  def __init__(self, epochs, lr, input_size):
    self.weight = np.ones(input_size)
    self.epochs = epochs
    self.lr = lr
    self.bias = 0.0

  def predict(self, x_test):
    a = self.bias
    for i in range(len(x_test)):
      a += self.weight[i] * x_test[i]
    return 1 if a>=0 else 0

  def train(self, train_data):
    for i in range(self.epochs):
      for x_train, y_train in train_data:
        predicted = self.predict(x_train)
        error = y_train - predicted
        self.bias += self.lr * error
        for j in range(len(self.weight)):
          self.weight[j] += self.lr * error * x_train[j]
```

```
perceptron = Perceptron(1000, 0.001, 8)
```

```
perceptron.train([([0,0,0,0,0,0,0,1], 0), ([0,0,0,0,0,0,1,0], 1),
([0,0,0,0,0,0,1,1], 0), ([0,0,0,0,0,1,0,0], 1), ([0,0,0,0,0,1,0,1], 0),
([0,0,0,0,0,1,1,0], 1), ([0,0,0,0,0,1,1,1], 0), ([0,0,0,0,1,0,0,0], 1),
([0,0,0,0,1,0,0,1], 0), ([0,0,0,0,1,0,1,0], 1), ([0,0,0,0,1,0,1,1], 0)])
```

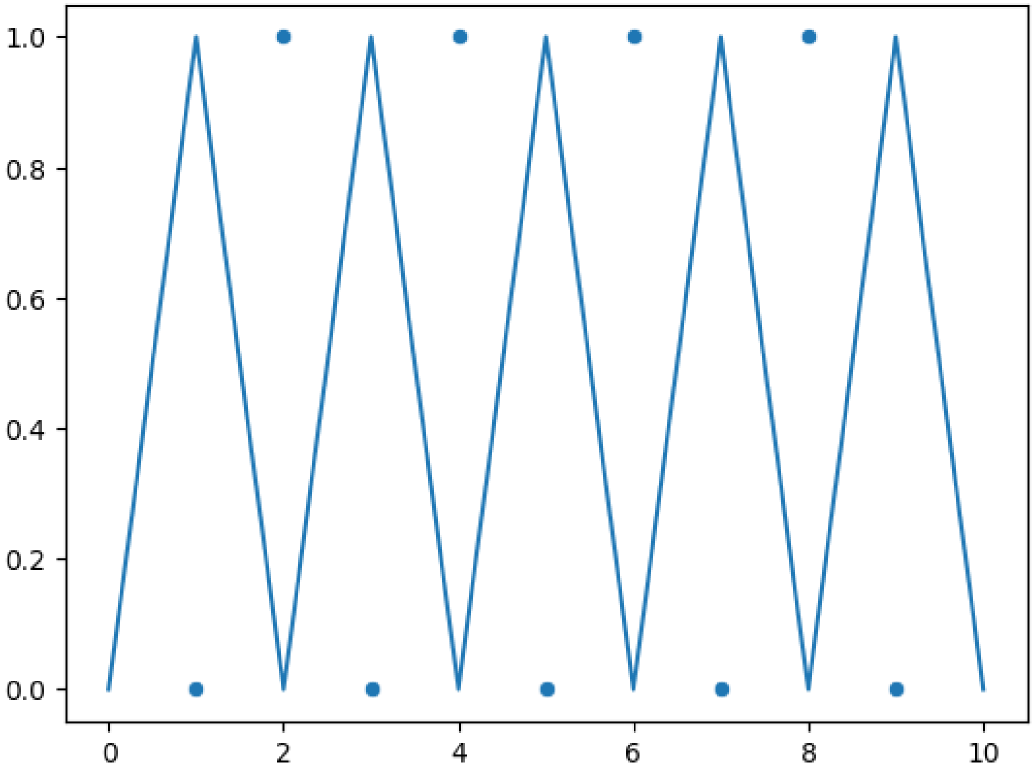
```
predictA = [ perceptron.predict(X_test) for X_test, i in [([0,0,0,0,0,0,0,1],
0), ([0,0,0,0,0,0,1,0], 1), ([0,0,0,0,0,0,1,1], 0), ([0,0,0,0,0,1,0,0], 1),
([0,0,0,0,0,1,0,1], 0), ([0,0,0,0,0,1,1,0], 1), ([0,0,0,0,0,1,1,1], 0),
([0,0,0,0,1,0,0,0], 1), ([0,0,0,0,1,0,0,1], 0), ([0,0,0,0,1,0,1,0], 1),
([0,0,0,0,1,0,1,1], 0)]]
```

```
perceptron.predict()
```

```
1
```

```
sns.scatterplot(x = [1, 2, 3, 4,5 ,6,7,8,9], y =[0, 1, 0, 1, 0, 1, 0, 1, 0])
plt.plot(predictA)
```

```
[<matplotlib.lines.Line2D at 0x7f34580cc820>]
```



```
a = [int(i) for i in bin(21)[2:]]
A = [0 for _ in range(8-len(a)) ] + a
A
```

```
[0, 0, 0, 1, 0, 1, 0, 1]
```

#Class For ASCII Input

```
class Perceptron():
  def __init__(self, epochs, lr, input_size):
    self.weight = np.ones(input_size)
    self.epochs = epochs
    self.lr = lr
    self.bias = 0.0

  def predict(self, x_test):
    a = self.bias
    x_test = self.binary(x_test)
    for i in range(len(x_test)):
      a += self.weight[i] * x_test[i]
    return 1 if a>=0 else 0

  def binary(self, x):
    a = [int(i) for i in bin(x)[2:]]
    A = [0 for _ in range(8-len(a)) ] + a
    return A

  def train(self, train_data):
    for i in range(self.epochs):
      for x_train, y_train in train_data:
        # print(x_train)
        # print(x_train)
        predicted = self.predict(x_train)
        error = y_train - predicted
        x_train = self.binary(x_train)
        self.bias += self.lr * error
        for j in range(len(self.weight)):
          self.weight[j] += self.lr * error * x_train[j]


p = Perceptron(1000, 0.001, 8)

x_train = []
for i in range(1, 100):
  if i % 2 == 0:
    x_train.append((i, 1))
  else:
    x_train.append((i, 0))

p.train(x_train)

p.weight
```

```
array([ 0.782,  0.781,  0.288,  0.288,  0.205,  0.183,  0.181, -1.749])
```

```
p.bias
```

```
-0.18000000000000005
```

```
predictions = [(i, p.predict(i)) for i in range(1, 13)]
```

```
predictions
```

Output:

```
Even odd numbers are:

[(1, 0),
 (2, 1),
 (3, 0),
 (4, 1),
 (5, 0),
 (6, 1),
 (7, 0),
 (8, 1),
 (9, 0),
 (10, 1),
 (11, 0),
 (12, 1),
 (13, 0)]
```