DEEP LEARNING

EXPERIMENT 1

Aim :

To demonstrate Confusion Matrix using python

Program :

import numpy as np

from sklearn.metrics import confusion\_matrix

import seaborn as sns

import matplotlib.pyplot as plt

actual=np.array(['Dog','Dog','Dog','Not Dog','Dog','Not Dog','Dog','Dog','Not Dog','Not Dog'])

predicted=np.array(['Dog','Not Dog','Dog','Not Dog','Dog','Dog','Dog','Dog','Not Dog','Not Dog'])

cm=confusion\_matrix(actual,predicted)

sns.heatmap(cm,annot=True,fmt='g',xticklabels=['Dog','Not Dog'], yticklabels=['Dog','Not Dog'], cmap='pink')

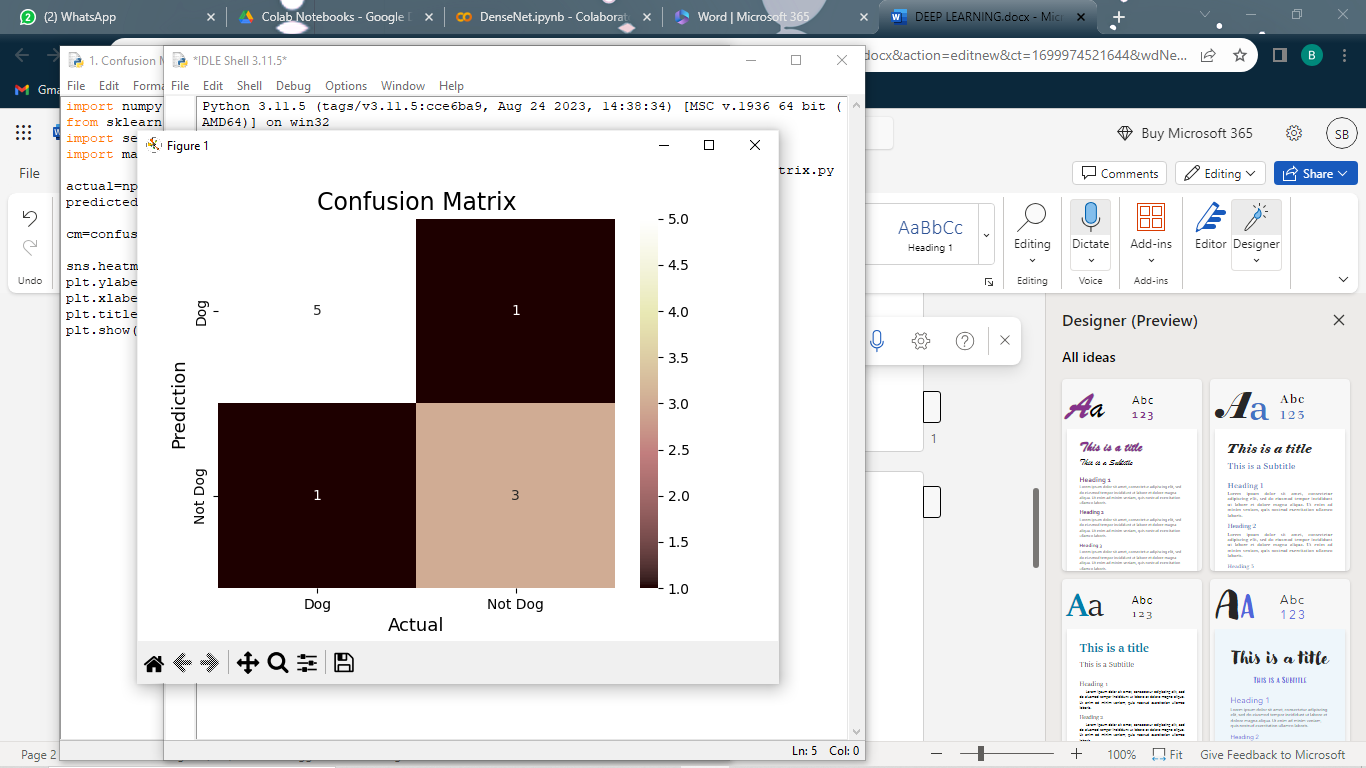
plt.ylabel("Prediction",fontsize=13)

plt.xlabel("Actual",fontsize=13)

plt.title("Confusion Matrix",fontsize=17)

plt.show()

Output :



EXPERIMENT 2 :

Aim :

Verifying the performance of a multi class confusion matrix by using choosen database with python code.

Program :

from sklearn.datasets import load\_breast\_cancer

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

from sklearn.tree import DecisionTreeClassifier

from sklearn.metrics import confusion\_matrix, accuracy\_score,precision\_score,recall\_score, f1\_score

import matplotlib.pyplot as plt

import seaborn as sns

X,y=load\_breast\_cancer(return\_X\_y=True)

x\_train,x\_test,y\_train,y\_test=train\_test\_split(X,y,test\_size=0.25, random\_state=23)

tree=DecisionTreeClassifier()

tree.fit(x\_train,y\_train)

y\_pred=tree.predict(x\_test)

cm=confusion\_matrix(y\_test,y\_pred)

accuracy=accuracy\_score(y\_test,y\_pred)

print('Accuracy : ',accuracy)

precision=precision\_score(y\_test,y\_pred)

print("Precision : ",precision)

recall=recall\_score(y\_test,y\_pred)

print("Recall : ",recall)

F1\_score=f1\_score(y\_test,y\_pred)

print("F1-score",F1\_score)

sns.heatmap(cm,annot=True,cmap='pink',xticklabels=['malignant','benign'],yticklabels=['malignant','benign'])

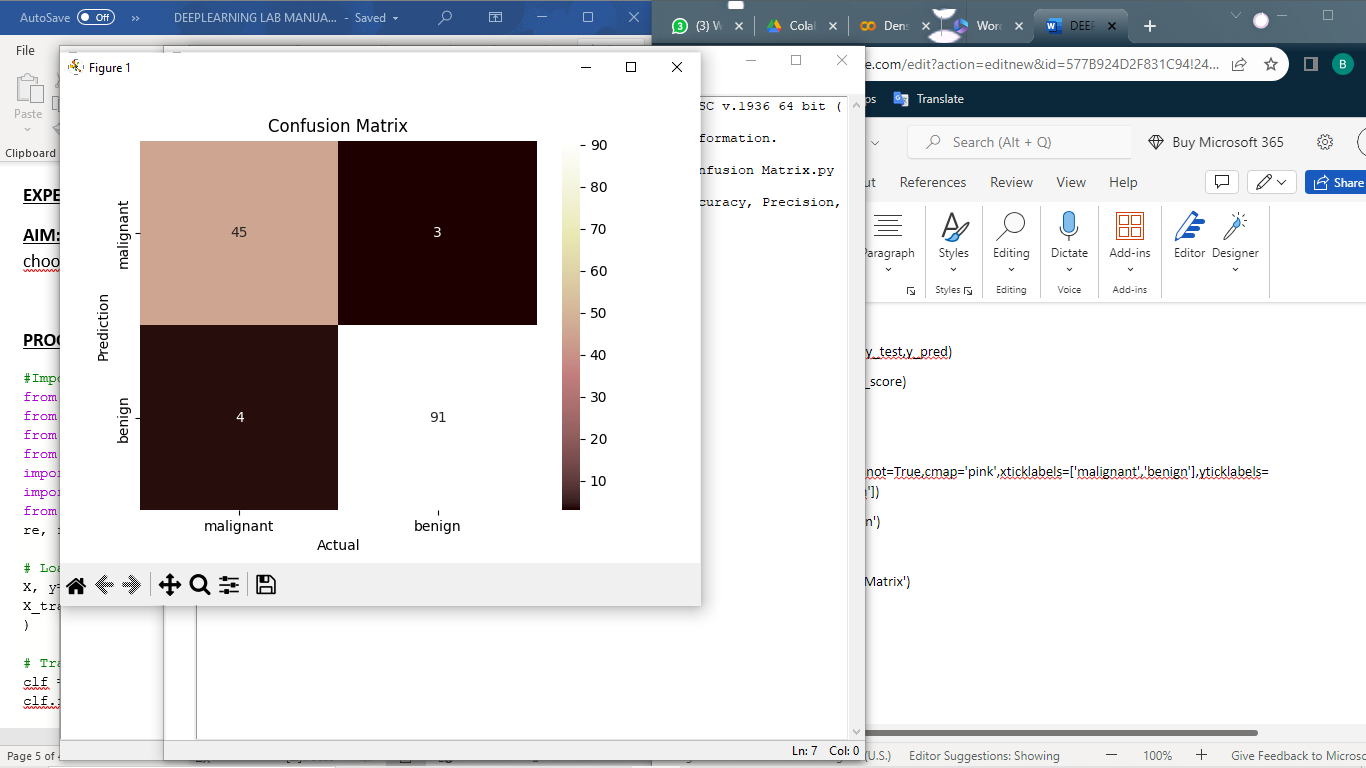
plt.ylabel('Prediction')

plt.xlabel("Actual")

plt.title('Confusion Matrix')

plt.show()

Output :



Accuracy : 0.951048951048951

Precision : 0.9680851063829787

Recall : 0.9578947368421052

F1-score 0.962962962962963

EXPERIMENT 3 :

Aim :

Verifying the performance of a over fitting by using choosen database with python code.

Program :

import numpy as np

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

from sklearn.linear\_model import LinearRegression

from sklearn.metrics import mean\_squared\_error

import matplotlib.pyplot as plt

np.random.seed(42)

x=2\*np.random.rand(100,1)

y=4+3\*x+np.random.randn(100,1)

x\_train,x\_test,y\_train,y\_test=train\_test\_split(x,y,test\_size=0.2,random\_state=42)

model=LinearRegression()

model.fit(x\_train,y\_train)

y\_train\_pred=model.predict(x\_train)

y\_test\_pred=model.predict(x\_test)

mse\_train=mean\_squared\_error(y\_train,y\_train\_pred)

mse\_test=mean\_squared\_error(y\_test,y\_test\_pred)

print(mse\_train)

print(mse\_test)

plt.scatter(x\_test,y\_test, color='yellow',label='Actual')

plt.plot(x\_test,y\_test\_pred,color='pink',label='Prediction')

plt.title('Linear Regression')

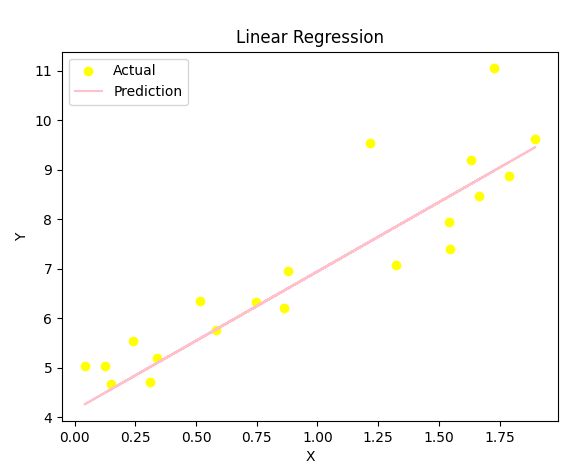
plt.xlabel('X')

plt.ylabel('Y')

plt.legend()

plt.show()

Output :



0.8476788564209705

0.6536995137170021

EXPERIMENT 4 :

Aim :

To demonstrate the performance of a linear regression by using choosen database with python code.

Program :

import numpy as np

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

from sklearn.linear\_model import LinearRegression

from sklearn.metrics import mean\_squared\_error

import matplotlib.pyplot as plt

np.random.seed(42)

x=2\*np.random.rand(100,1)

y=4+3\*x+np.random.randn(100,1)

x\_train,x\_test,y\_train,y\_test=train\_test\_split(x,y,test\_size=0.2, random\_state=42)

model=LinearRegression()

model.fit(x\_train,y\_train)

y\_pred=model.predict(x\_test)

mse=mean\_squared\_error(y\_test,y\_pred)

print(mse)

plt.scatter(x\_test,y\_test,color='pink',label='Actual')

plt.plot(x\_test,y\_pred,color='yellow',label='Prediction')

plt.title('Linear Regression')

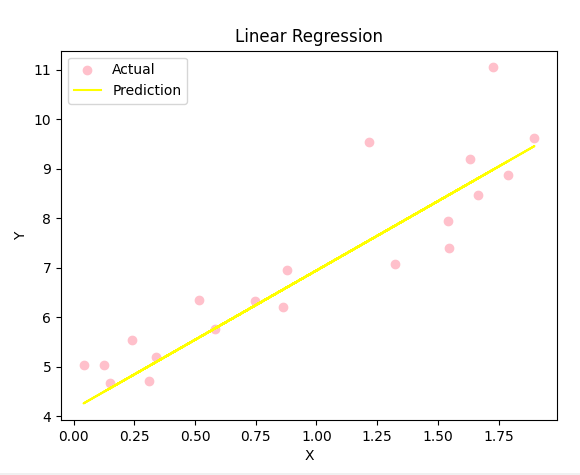
plt.xlabel('X')

plt.ylabel('Y')

plt.legend()

plt.show()

Output :



0.6536995137170021

EXPERIMENT 5 :

Aim :

To demonstrate the performance of a logistic regression by using chosen database with python code.

Program :

import numpy as np

import matplotlib.pyplot as plt

def sigmoid(z):

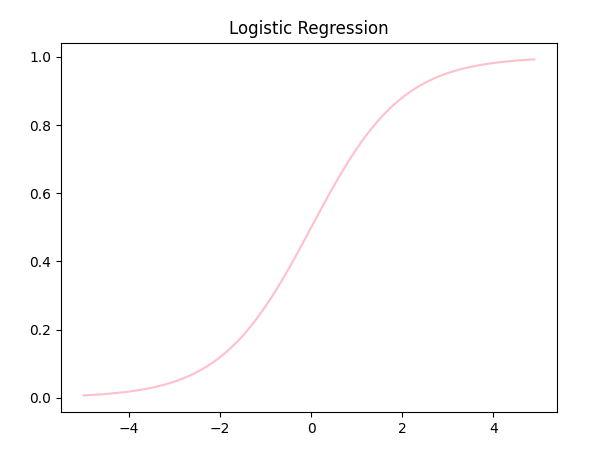
return 1/(1+np.exp(-z))

plt.plot(np.arange(-5,5,0.1),sigmoid(np.arange(-5,5,0.1)), color='pink')

plt.title('Logistic Regression')

plt.show()

Output :



EXPERIMENT 6(A) :

Aim :

Finding the accuracyvalue of iris data set using KNN algorithm

Program :

import numpy as np

import pandas as pd

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

from sklearn.datasets import load\_iris

X,y=load\_iris(return\_X\_y=True)

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.20, random\_state = 42)

from sklearn.neighbors import KNeighborsClassifier

classifier = KNeighborsClassifier(n\_neighbors = 5, metric = 'minkowski', p = 2)

classifier.fit(X\_train, y\_train)

from sklearn.metrics import confusion\_matrix, accuracy\_score

y\_pred = classifier.predict(X\_test)

cm = confusion\_matrix(y\_test, y\_pred)

print(cm)

accuracy=accuracy\_score(y\_test, y\_pred)

print(accuracy)

Output :

[[10 0 0]

[ 0 9 0]

[ 0 0 11]]

1.0

EXPERIMENT 6(B) :

Aim :

Finding accuracy value of iris data set using Naïve Bayes algorithm.

Program :

import numpy as np

import pandas as pd

from sklearn.datasets import load\_iris

X,y=load\_iris(return\_X\_y=True)

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

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.25, random\_state = 0)

from sklearn.naive\_bayes import GaussianNB

classifier = GaussianNB()

classifier.fit(X\_train, y\_train)

from sklearn.metrics import confusion\_matrix, accuracy\_score

y\_pred = classifier.predict(X\_test)

cm = confusion\_matrix(y\_test, y\_pred)

print(cm)

accuracy=accuracy\_score(y\_test, y\_pred)

print(accuracy)

Output :

[[13 0 0]

[ 0 16 0]

[ 0 0 9]]

1.0

EXPERIMENT 6(C) :

Aim :

Finding accuracy value of iris data set using Logistic Regression algorithm.

Program :

import numpy as np

import pandas as pd

dataset = pd.read\_csv("D:\\Desktop\\Bhavadharani-studies\\Fundamental of machine learning\\breastcancer.csv")

X = dataset.iloc[:, :-1].values

y = dataset.iloc[:, -1].values

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

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.30, random\_state = 2)

from sklearn.linear\_model import LogisticRegression

classifier = LogisticRegression(random\_state = 0)

classifier.fit(X\_train, y\_train)

from sklearn.metrics import confusion\_matrix, accuracy\_score

y\_pred = classifier.predict(X\_test)

cm = confusion\_matrix(y\_test, y\_pred)

print(cm)

accuracy=accuracy\_score(y\_test, y\_pred)

print(accuracy)

Output :

[[125 0]

[ 80 0]]

0.6097560975609756

EXPERIMENT 6(D) :

Aim :

Finding accuracy value of iris data set using Decision Tree algorithm.

Program :

import numpy as np

import pandas as pd

dataset = pd.read\_csv("D:\\Desktop\\Bhavadharani-studies\\Fundamental of machine learning\\breastcancer.csv")

X = dataset.iloc[:, :-1].values

y = dataset.iloc[:, -1].values

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

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.30, random\_state = 2)

from sklearn.tree import DecisionTreeClassifier

classifier = DecisionTreeClassifier(random\_state = 0)

classifier.fit(X\_train, y\_train)

from sklearn.metrics import confusion\_matrix, accuracy\_score

y\_pred = classifier.predict(X\_test)

cm = confusion\_matrix(y\_test, y\_pred)

print(cm)

accuracy=accuracy\_score(y\_test, y\_pred)

print(accuracy)

Output :

[[114 11]

[ 11 69]]

0.8926829268292683

EXPERIMENT 6(E) :

Aim :

Finding accuracy value of iris data set using Support Vector Machine algorithm.

Program :

import numpy as np

import pandas as pd

dataset = pd.read\_csv("D:\\Desktop\\Bhavadharani-studies\\Fundamental of machine learning\\breastcancer.csv")

X = dataset.iloc[:, :-1].values

y = dataset.iloc[:, -1].values

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

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.30, random\_state = 2)

from sklearn.svm import SVC

classifier = SVC(random\_state = 0)

classifier.fit(X\_train, y\_train)

from sklearn.metrics import confusion\_matrix, accuracy\_score

y\_pred = classifier.predict(X\_test)

cm = confusion\_matrix(y\_test, y\_pred)

print(cm)

accuracy=accuracy\_score(y\_test, y\_pred)

print(accuracy)

Output :

[[125 0]

[ 80 0]]

0.6097560975609756

EXPERIMENT 6(F) :

Aim :

Finding accuracy value of iris data set using Random forest algorithm.

Program :

import numpy as np

import pandas as pd

dataset = pd.read\_csv("D:\\Desktop\\Bhavadharani-studies\\Fundamental of machine learning\\breastcancer.csv")

X = dataset.iloc[:, :-1].values

y = dataset.iloc[:, -1].values

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

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.30, random\_state = 2)

from sklearn.ensemble import RandomForestClassifier

classifier = RandomForestClassifier(random\_state = 0)

classifier.fit(X\_train, y\_train)

from sklearn.metrics import confusion\_matrix, accuracy\_score

y\_pred = classifier.predict(X\_test)

cm = confusion\_matrix(y\_test, y\_pred)

print(cm)

accuracy=accuracy\_score(y\_test, y\_pred)

print(accuracy)

Output :

[[118 7]

[ 1 79]]

0.9609756097560975

EXPERIMENT 7(A) :

Aim :

To demonstrate gradient descent using python.

Program :

import numpy as np

import matplotlib.pyplot as plt

np.random.seed(42)

X=2\*np.random.rand(100,1)

y=4+3\*X+np.random.randn(100,1)

x\_b=np.c\_[np.ones((100,1)),X]

learning\_rate=0.01

n\_iterations=1000

theta=np.random.randn(2,1)

for iteration in range(n\_iterations):

gradients=2/100\*x\_b.T.dot(x\_b.dot(theta)-y)

theta=theta-learning\_rate\*gradients

print(theta)

plt.scatter(X,y,label='Data')

plt.plot(X,x\_b.dot(theta),color='pink',label='Linear Regression')

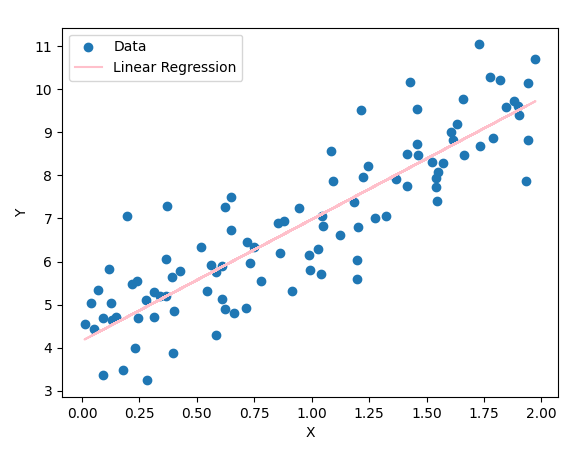
plt.xlabel('X')

plt.ylabel('Y')

plt.legend()

plt.show()

Output :



[[4.15809376]

[2.8204434 ]]

EXPERIMENT 7(B) :

Aim :

To demonstrate gradient descent using python.

Program :

import numpy as np

import matplotlib.pyplot as plt

np.random.seed(42)

X=2\*np.random.rand(100,1)

y=4+3\*X+np.random.randn(100,1)

x\_b=np.c\_[np.ones((100,1)),X]

learning\_rate=0.01

n\_iterations=1000

theta=np.random.randn(2,1)

for iteration in range(n\_iterations):

gradients=2/100\*x\_b.T.dot(x\_b.dot(theta)-y)

theta=theta-learning\_rate\*gradients

print(theta)

plt.scatter(X,y,label='Data')

plt.plot(X,x\_b.dot(theta),color='pink',label='Linear Regression')

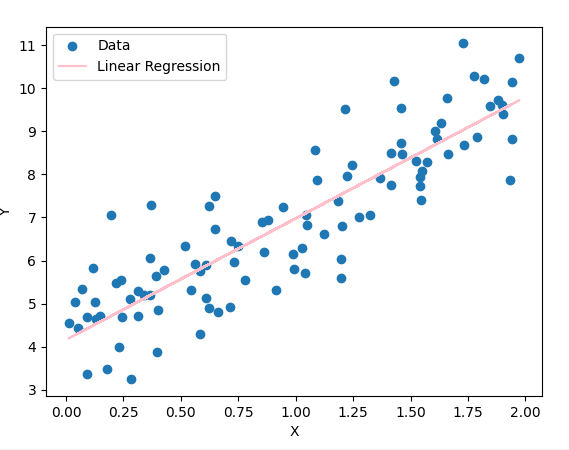
plt.xlabel('X')

plt.ylabel('Y')

plt.legend()

plt.show()

Output :



[[4.15809376]

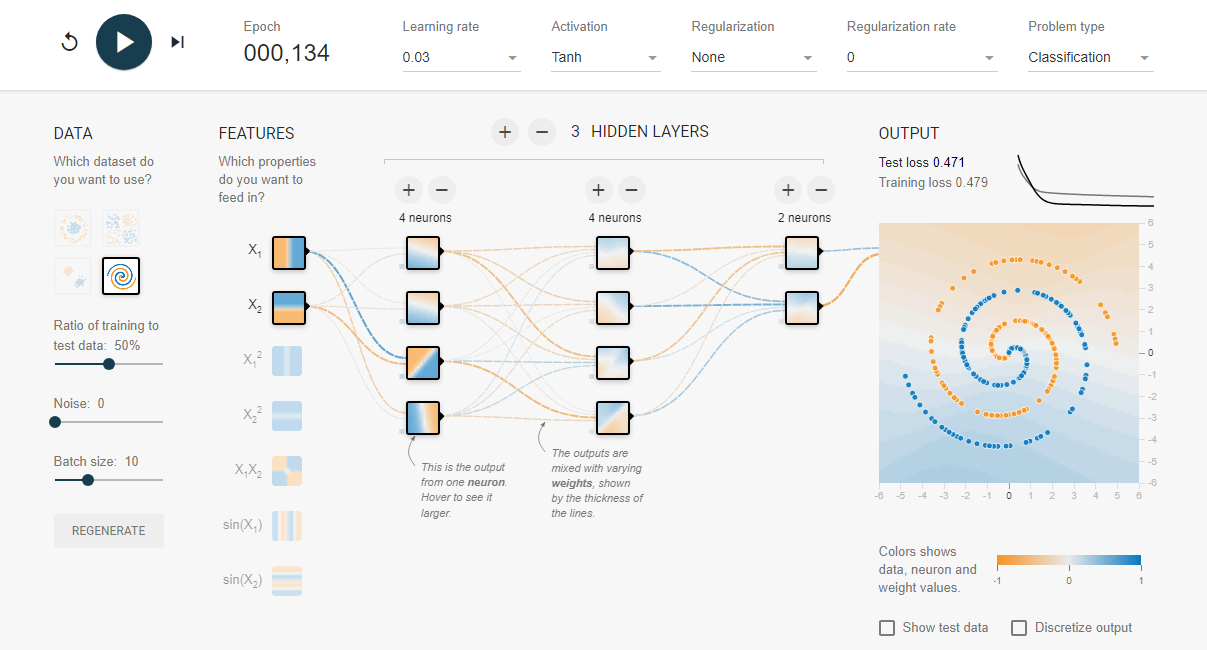
[2.8204434 ]]

EXPERIMENT 8(A) :

Aim :

Neural Network analysis using TANH activation.

Output :

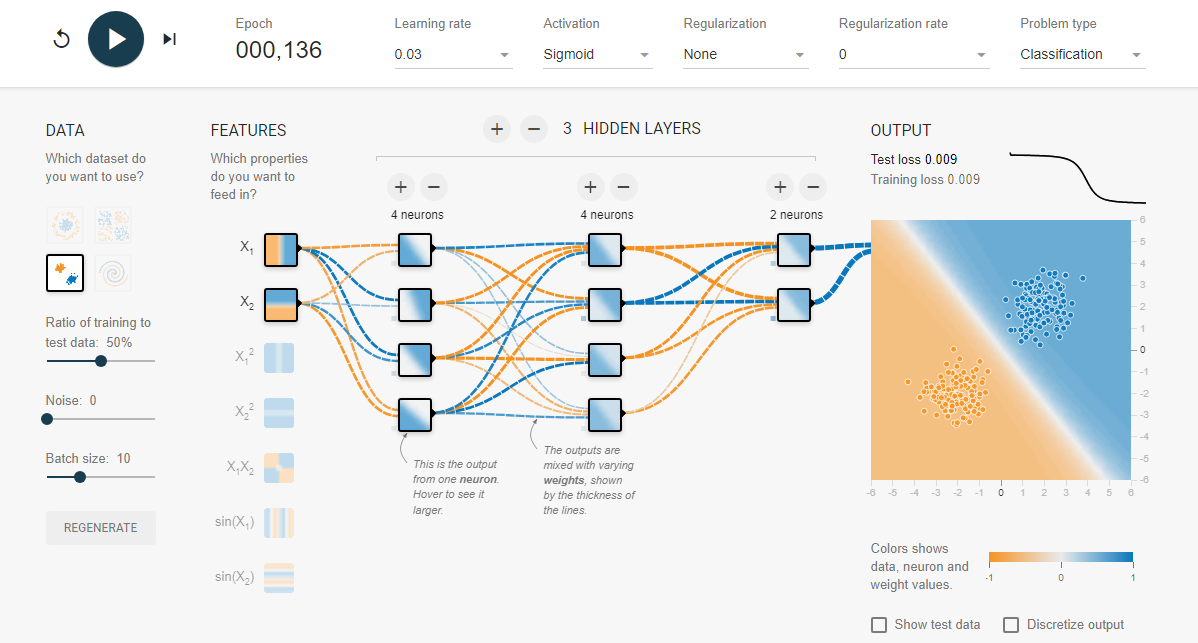


EXPERIMENT 8(B) :

Aim :

Neural Network analysis using SIGMOID activation.

Output :

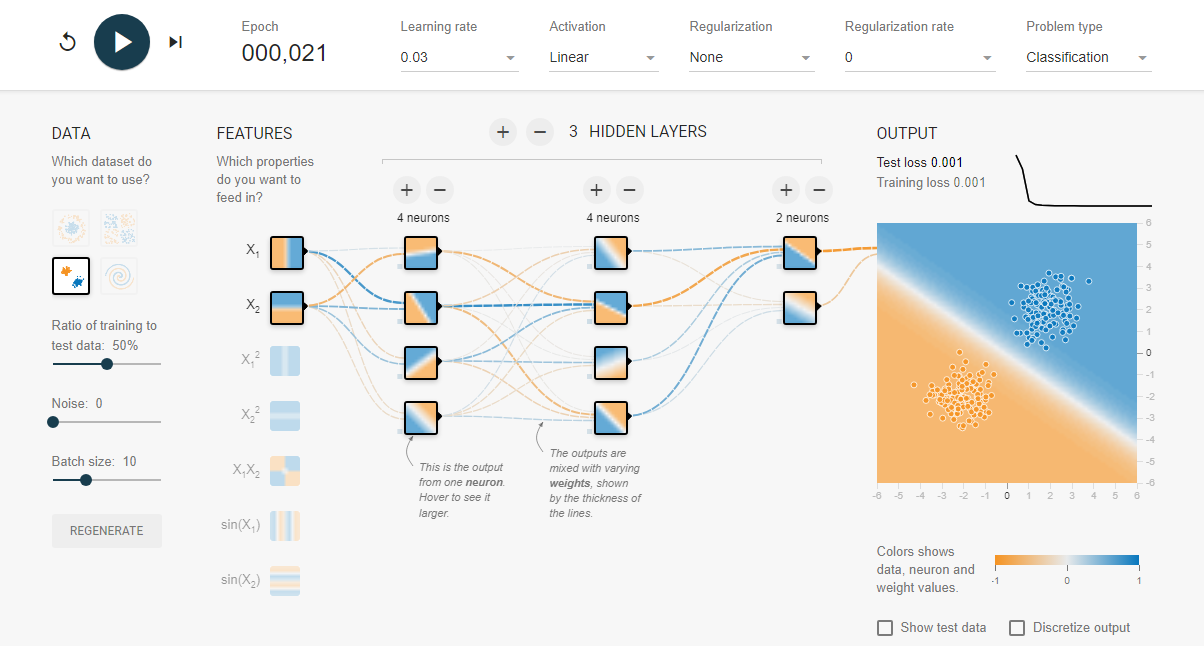


EXPERIMENT 8(C) :

Aim :

Neural Network analysis using LINEAR activation.

Output :

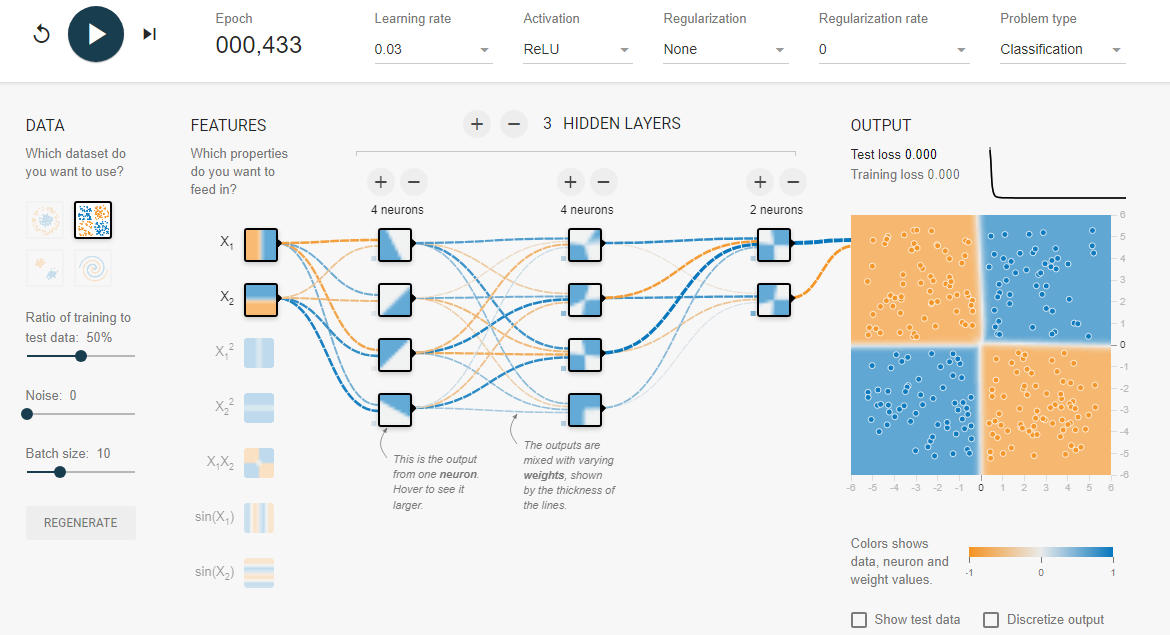


EXPERIMENT 8(D) :

Aim :

Neural Network analysis usinng ReLU activation.

Output :



EXPERIMENT 9 :

Aim :

To demonstrate linear separability using python code.

Program :

import numpy as np

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

from sklearn.linear\_model import LinearRegression

from sklearn.metrics import mean\_squared\_error

import matplotlib.pyplot as plt

np.random.seed(42)

x=2\*np.random.rand(100,1)

y=4+3\*x+np.random.randn(100,1)

x\_train,x\_test,y\_train,y\_test=train\_test\_split(x,y,test\_size=0.2, random\_state=42)

model=LinearRegression()

model.fit(x\_train,y\_train)

y\_pred=model.predict(x\_test)

mse=mean\_squared\_error(y\_test,y\_pred)

print(mse)

plt.scatter(x\_test,y\_test,color='pink',label='Actual')

plt.plot(x\_test,y\_pred,color='yellow',label='Prediction')

plt.title('Linear Regression')

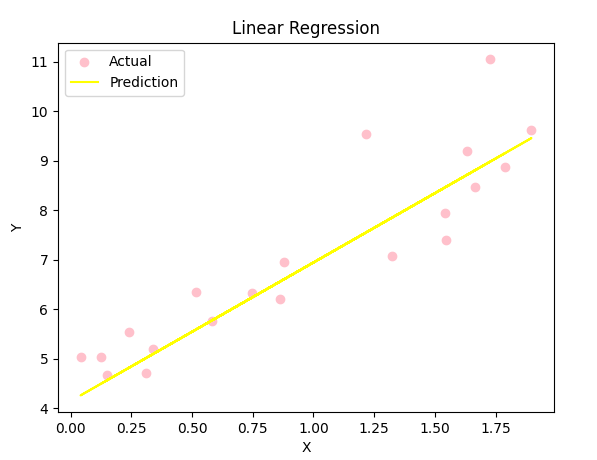
plt.xlabel('X')

plt.ylabel('Y')

plt.legend()

plt.show()

Output :



0.6536995137170021