In [1]:

```
from sklearn.linear_model import Perceptron
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.model_selection import cross_validate
from sklearn.neural_network import MLPClassifier
from sklearn.svm import SVC
from sklearn.preprocessing import StandardScaler
from sklearn.multiclass import OneVsRestClassifier
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix
```

In [2]:

```
import pandas as pd
import numpy as np
```

In [ ]:

```
df = pd.read_csv("/content/train.csv")
```

In [ ]:

```
df_test = pd.read_csv("/content/dev.csv")
```

In [ ]:

```
df.columns = ["x", "y", "label"]
```

In [ ]:

```
df_test.columns = ["x", "y", "label"]
```

In [ ]:

```
df.head()
```

Out[ ]:

	x	y	label
0	-9.880677	-9.137125	2.0
1	6.090265	-0.531222	1.0
2	-8.018714	-9.678179	2.0
3	1.735207	-7.480774	3.0
4	1.141463	-8.783495	3.0

In [ ]:

```
df_test.head()
```

Out[ ]:

	x	y	label
0	-10.216441	-8.062278	2.0
1	-8.846681	-9.217024	2.0
2	-9.874947	-8.782837	2.0
3	-9.530322	-0.116388	0.0
4	7.199072	-1.237567	1.0

In [ ]:

```
df_0_1 = df[(df.label == 0.0) | (df.label == 1.0) ]
```

In [ ]:

```
df_1_2 = df[(df.label == 1.0) | (df.label == 2.0)]
```

In [ ]:

```
df_0_2 = df[(df.label == 0.0) | (df.label == 2.0)]
```

In [ ]:

```
df_0_3 = df[(df.label == 0.0) | (df.label == 3.0) ]
```

In [ ]:

```
df_1_3 = df[(df.label == 1.0) | (df.label == 3.0)]
```

In [ ]:

```
df_2_3 = df[(df.label == 2.0) | (df.label == 3.0)]
```

In [ ]:

```
df_0_1_test = df_test[(df_test.label == 0.0) | (df_test.label == 1.0) ]
```

In [ ]:

```
df_1_2_test = df_test[(df_test.label == 1.0) | (df_test.label == 2.0)]
```

In [ ]:

```
df_0_2_test = df_test[(df_test.label == 0.0) | (df_test.label == 2.0)]
```

In [ ]:

```
df_0_3_test = df_test[(df_test.label == 0.0) | (df_test.label == 3.0) ]
```

In [ ]:

```
df_1_3_test = df_test[(df_test.label == 1.0) | (df_test.label == 3.0)]
```

In [ ]:

```
df_2_3_test = df_test[(df_test.label == 2.0) | (df_test.label == 3.0)]
```

Perceptron for every pair of classes

In [3]:

```
def plot_pairs(best_model, df):
    X = df[['x','y']].values
    xmin, xmax = np.min(X[:,0]) - 1, np.max(X[:,0]) + 1
    ymin, ymax = np.min(X[:,1]) - 1, np.max(X[:,1]) + 1

    x = np.linspace(xmin,xmax,100)
    y = np.linspace(ymin,ymax,100)

    xx, yy = np.meshgrid(x,y)

    Z = np.stack((xx.ravel(),yy.ravel()),axis = 1)

    Z = scaler.transform(Z)
    pred = best_model.predict(Z).reshape(100,100)
    # print(pred.shape)
    plt.contourf(xx,yy,pred)
    plt.scatter(X[:,0],X[:,1],c = df.label, edgecolors= 'k')
    plt.xlabel("X")
    plt.ylabel("Y")
    i,j = df.label.unique()
    plt.title(f'Perceptron model for classes {i} v/s {j}')
    plt.show()
```

In [4]:

```
def plot_pairs_svm_no_scaling(best_model, df):
    sv = best_model.support_vectors_
    X = df[['x','y']].values
    xmin, xmax = np.min(X[:,0]) - 1, np.max(X[:,0]) + 1
    ymin, ymax = np.min(X[:,1]) - 1, np.max(X[:,1]) + 1

    x = np.linspace(xmin,xmax,100)
    y = np.linspace(ymin,ymax,100)

    xx, yy = np.meshgrid(x,y)

    Z = np.stack((xx.ravel(),yy.ravel()),axis = 1)

    Z = scaler.transform(Z)
    pred = best_model.predict(Z).reshape(100,100)
    # print(pred.shape)

    plt.contourf(xx,yy,pred)
    plt.scatter(X[:,0],X[:,1],c = df.label, edgecolors= 'k', alpha= 0.5)
    plt.scatter(sv[:, 0], sv[:,1], c = "red")
    plt.xlabel("X")
    plt.ylabel("Y")
    i,j = df.label.unique()
    plt.title(f'Linear SVM for classes {i} v/s {j}')
    plt.show()
```

In [5]:

```
def plot_pairs_svm(best_model, df):
    sv = best_model.support_vectors_
    scaler = StandardScaler()
    scaler.fit(df.drop(columns=['label']))
    X = df[['x','y']].values
    xmin, xmax = np.min(X[:,0]) - 1, np.max(X[:,0]) + 1
    ymin, ymax = np.min(X[:,1]) - 1, np.max(X[:,1]) + 1

    x = np.linspace(xmin,xmax,100)
    y = np.linspace(ymin,ymax,100)

    xx, yy = np.meshgrid(x,y)

    Z = np.stack((xx.ravel(),yy.ravel()),axis = 1)

    Z = scaler.transform(Z)
    sv = scaler.inverse_transform(sv)
    pred = best_model.predict(Z).reshape(100,100)
    # print(pred.shape)

    plt.contourf(xx,yy,pred)
    plt.scatter(X[:,0],X[:,1],c = df.label, edgecolors= 'k', alpha=0.5)
    plt.scatter(sv[:, 0], sv[:,1], c = "red")
    plt.xlabel("X")
    plt.ylabel("Y")
    i,j = df.label.unique()
    plt.title(f'SVM for classes {i} v/s {j}')
    plt.show()
```

0 v/s 1

In [ ]:

```
hyperparams = [0.001, 0.01, 0.1, 1, 10]
```

In [ ]:

```
for i in hyperparams:
    model = Perceptron(eta0=i)
    cv_results = cross_validate(model, df_0_1.drop(columns= ["label"]), df_0_1.label, cv
=3)
    print("Cross Validation Val Score for learning rate = ", i, "is",np.mean(cv_results[
'test_score']))
    model = Perceptron(eta0=i)
    train_acc = model.fit(df_0_1.drop(columns= ["label"]), df_0_1.label).score(df_0_1.dr
op(columns= ["label"]), df_0_1.label)
    print("Train accuracy for learning rate = ", i, "is ", train_acc)
```

```
Cross Validation Val Score for learning rate =  0.001 is 1.0
Train accuracy for learning rate =  0.001 is  1.0
Cross Validation Val Score for learning rate =  0.01 is 1.0
Train accuracy for learning rate =  0.01 is  1.0
Cross Validation Val Score for learning rate =  0.1 is 1.0
Train accuracy for learning rate =  0.1 is  1.0
Cross Validation Val Score for learning rate =  1 is 1.0
Train accuracy for learning rate =  1 is  1.0
Cross Validation Val Score for learning rate =  10 is 1.0
Train accuracy for learning rate =  10 is  1.0
```

In [ ]:

```
best_model  = Perceptron(eta0=0.001).fit(df_0_1.drop(columns= ["label"]), df_0_1.label)
```

In [ ]:

```
test_acc = best_model.score(df_0_1_test.drop(columns= ["label"]), df_0_1_test.label)
print("Accuracy on the test set is ", test_acc)
```

```
Accuracy on the test set is  1.0
```

1 v/s 2

In [ ]:

```
for i in hyperparams:
    model = Perceptron(eta0=i)
    cv_results = cross_validate(model, df_1_2.drop(columns= ["label"]), df_1_2.label, cv
=3)
    print("Cross Validation Val Score for learning rate = ", i, "is",np.mean(cv_results[
'test_score']))
    model = Perceptron(eta0=i)
    train_acc = model.fit(df_1_2.drop(columns= ["label"]), df_1_2.label).score(df_1_2.dr
op(columns= ["label"]), df_1_2.label)
    print("Train accuracy for learning rate = ", i, "is ", train_acc)
```

```
Cross Validation Val Score for learning rate =  0.001 is 1.0
Train accuracy for learning rate =  0.001 is  1.0
Cross Validation Val Score for learning rate =  0.01 is 1.0
Train accuracy for learning rate =  0.01 is  1.0
Cross Validation Val Score for learning rate =  0.1 is 1.0
Train accuracy for learning rate =  0.1 is  1.0
Cross Validation Val Score for learning rate =  1 is 1.0
Train accuracy for learning rate =  1 is  1.0
Cross Validation Val Score for learning rate =  10 is 1.0
Train accuracy for learning rate =  10 is  1.0
```

In [ ]:

```
model = Perceptron(eta0=0.001)
test_acc = model.fit(df_1_2.drop(columns= ["label"]), df_1_2.label).score(df_1_2_test.dr
op(columns= ["label"]), df_1_2_test.label)
print("Accuracy on the test set is ", test_acc)
```

```
Accuracy on the test set is  1.0
```

0 v/s 2

In [ ]:

```
for i in hyperparams:
    model = Perceptron(eta0=i)
    cv_results = cross_validate(model, df_0_2.drop(columns= ["label"]), df_0_2.label, cv
=3)
    print("Cross Validation Val Score for learning rate = ", i, "is",np.mean(cv_results[
'test_score']))
    model = Perceptron(eta0=i)
    train_acc = model.fit(df_0_2.drop(columns= ["label"]), df_0_2.label).score(df_0_2.dr
op(columns= ["label"]), df_0_2.label)
    print("Train accuracy for learning rate = ", i, "is ", train_acc)
```

```
Cross Validation Val Score for learning rate =  0.001 is 1.0
Train accuracy for learning rate =  0.001 is  1.0
Cross Validation Val Score for learning rate =  0.01 is 1.0
Train accuracy for learning rate =  0.01 is  1.0
Cross Validation Val Score for learning rate =  0.1 is 1.0
Train accuracy for learning rate =  0.1 is  1.0
Cross Validation Val Score for learning rate =  1 is 1.0
Train accuracy for learning rate =  1 is  1.0
Cross Validation Val Score for learning rate =  10 is 1.0
Train accuracy for learning rate =  10 is  1.0
```

In [ ]:

```
model = Perceptron(eta0=0.001)
test_acc = model.fit(df_0_2.drop(columns= ["label"]), df_0_2.label).score(df_0_2_test.dr
op(columns= ["label"]), df_0_2_test.label)
print("Accuracy on the test set is ", test_acc)
```

```
Accuracy on the test set is  1.0
```

0 v/s 3

In [ ]:

```
for i in hyperparams:
    model = Perceptron(eta0=i)
    cv_results = cross_validate(model, df_0_3.drop(columns= ["label"]), df_0_3.label, cv
=3)
    print("Cross Validation Val Score for learning rate = ", i, "is",np.mean(cv_results[
'test_score']))
    model = Perceptron(eta0=i)
    train_acc = model.fit(df_0_3.drop(columns= ["label"]), df_0_3.label).score(df_0_3.dr
op(columns= ["label"]), df_0_3.label)
    print("Train accuracy for learning rate = ", i, "is ", train_acc)
```

```
Cross Validation Val Score for learning rate =  0.001 is 1.0
Train accuracy for learning rate =  0.001 is  1.0
Cross Validation Val Score for learning rate =  0.01 is 1.0
Train accuracy for learning rate =  0.01 is  1.0
Cross Validation Val Score for learning rate =  0.1 is 1.0
Train accuracy for learning rate =  0.1 is  1.0
Cross Validation Val Score for learning rate =  1 is 1.0
Train accuracy for learning rate =  1 is  1.0
Cross Validation Val Score for learning rate =  10 is 1.0
Train accuracy for learning rate =  10 is  1.0
```

In [ ]:

```
model = Perceptron(eta0=0.001)
```

```
test_acc = model.fit(df_0_3.drop(columns= ["label"]), df_0_3.label).score(df_0_3_test.dr
op(columns= ["label"]), df_0_3_test.label)
print("Accuracy on the test set is ", test_acc)
```

```
Accuracy on the test set is  1.0
```

1 v/s 3

In [ ]:

```
for i in hyperparams:
    model = Perceptron(eta0=i)
    cv_results = cross_validate(model, df_1_3.drop(columns= ["label"]), df_1_3.label, cv
=3)
    print("Cross Validation Val Score for learning rate = ", i, "is",np.mean(cv_results[
'test_score']))
    model = Perceptron(eta0=i)
    train_acc = model.fit(df_1_3.drop(columns= ["label"]), df_1_3.label).score(df_1_3.dr
op(columns= ["label"]), df_1_3.label)
    print("Train accuracy for learning rate = ", i, "is ", train_acc)
```

```
Cross Validation Val Score for learning rate =  0.001 is 1.0
Train accuracy for learning rate =  0.001 is  1.0
Cross Validation Val Score for learning rate =  0.01 is 1.0
Train accuracy for learning rate =  0.01 is  1.0
Cross Validation Val Score for learning rate =  0.1 is 1.0
Train accuracy for learning rate =  0.1 is  1.0
Cross Validation Val Score for learning rate =  1 is 1.0
Train accuracy for learning rate =  1 is  1.0
Cross Validation Val Score for learning rate =  10 is 1.0
Train accuracy for learning rate =  10 is  1.0
```

In [ ]:

```
model = Perceptron(eta0=0.001)
test_acc = model.fit(df_1_3.drop(columns= ["label"]), df_1_3.label).score(df_1_3_test.dr
op(columns= ["label"]), df_1_3_test.label)
print("Accuracy on the test set is ", test_acc)
```

```
Accuracy on the test set is  1.0
```

2 v/s 3

In [ ]:

```
for i in hyperparams:
    model = Perceptron(eta0=i)
    cv_results = cross_validate(model, df_2_3.drop(columns= ["label"]), df_2_3.label, cv
=3)
    print("Cross Validation Val Score for learning rate = ", i, "is",np.mean(cv_results[
'test_score']))
    model = Perceptron(eta0=i)
    train_acc = model.fit(df_2_3.drop(columns= ["label"]), df_2_3.label).score(df_2_3.dr
op(columns= ["label"]), df_2_3.label)
    print("Train accuracy for learning rate = ", i, "is ", train_acc)
```

```
Cross Validation Val Score for learning rate =  0.001 is 1.0
Train accuracy for learning rate =  0.001 is  1.0
Cross Validation Val Score for learning rate =  0.01 is 1.0
Train accuracy for learning rate =  0.01 is  1.0
Cross Validation Val Score for learning rate =  0.1 is 1.0
Train accuracy for learning rate =  0.1 is  1.0
Cross Validation Val Score for learning rate =  1 is 1.0
Train accuracy for learning rate =  1 is  1.0
Cross Validation Val Score for learning rate =  10 is 1.0
Train accuracy for learning rate =  10 is  1.0
```

In [ ]:

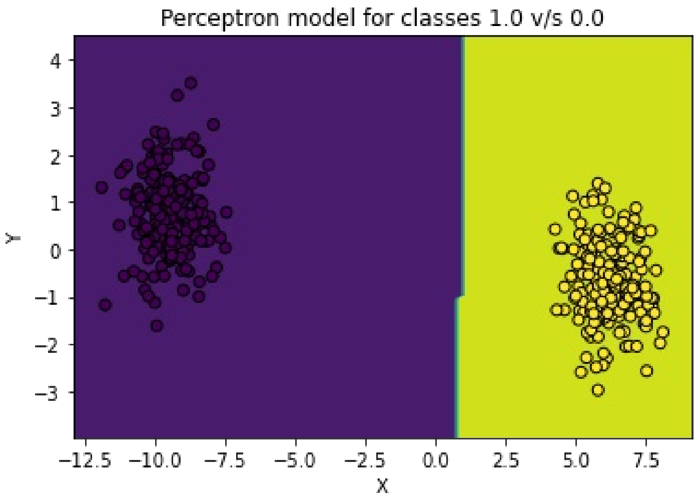
```
model = Perceptron(eta0=0.001)
test_acc = model.fit(df_2_3.drop(columns= ["label"]), df_2_3.label).score(df_2_3_test.dr
op(columns= ["label"]), df_2_3_test.label)
print("Accuracy on the test set is ", test_acc)
```

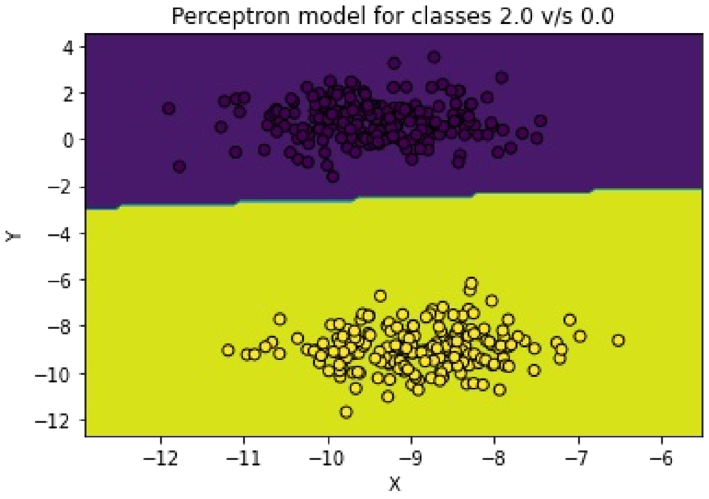
```
Accuracy on the test set is  1.0
```

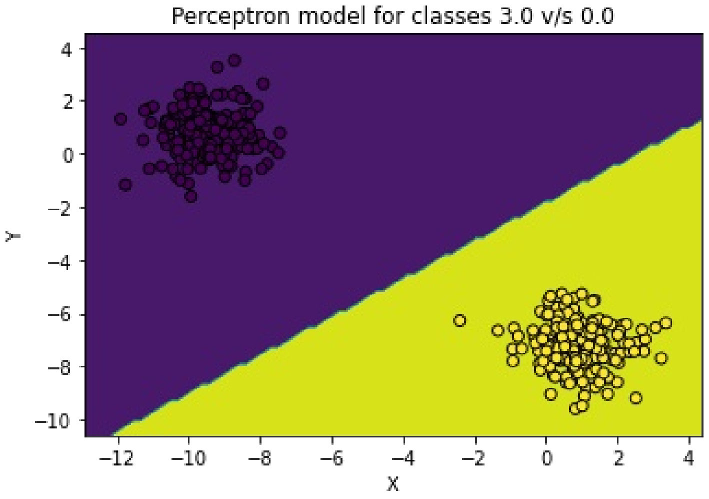
Plotting decision boundaries for pairwise classifiers

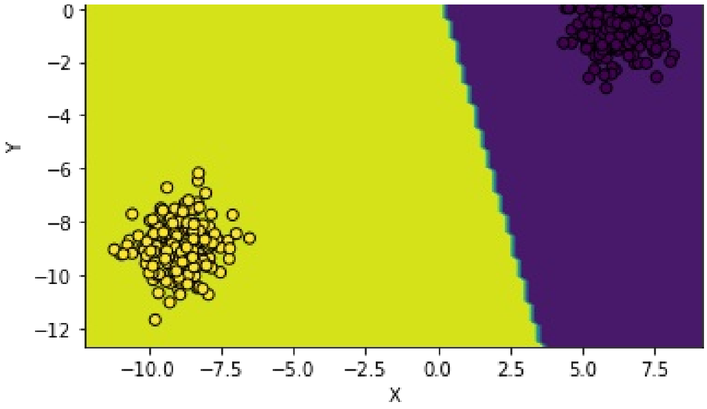
In [ ]:

```
for df in [df_0_1, df_0_2, df_0_3, df_1_2, df_1_3, df_2_3]:
    best_model = Perceptron(eta0=0.001).fit(df.drop(columns= ["label"]), df.label)
    plot_pairs(best_model, df)
```

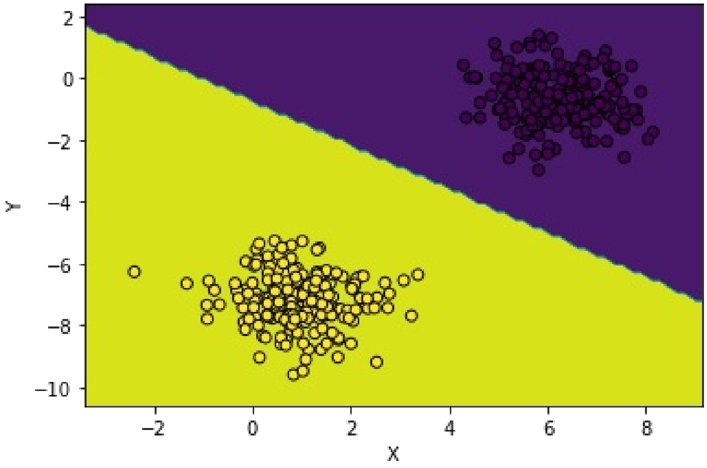




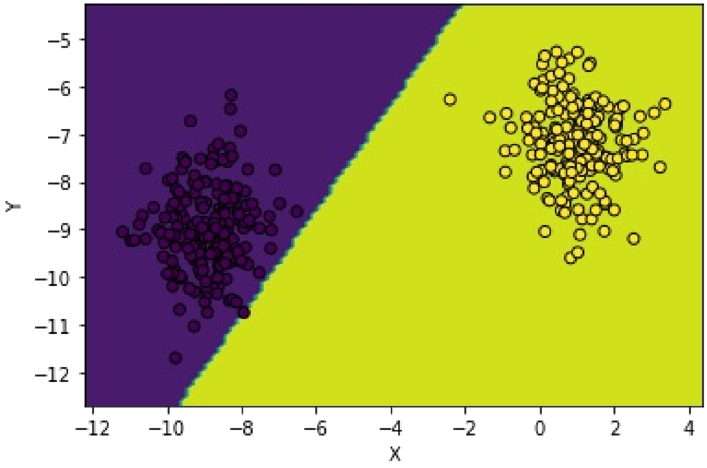




Perceptron model for classes 1.0 v/s 3.0



Perceptron model for classes 2.0 v/s 3.0



Multilayer feedforward neural network (MLFFNN) with a single hidden layer for all classes

In [ ]:

```
scaler = StandardScaler()
scaler.fit(df.drop(columns= ["label"]).values)
df_scaled = scaler.transform(df.drop(columns= ["label"]).values)
df_test_scaled = scaler.transform(df_test.drop(columns= ["label"]).values)
```

In [ ]:

```
num_neurons = [4, 5, 7]
```

In [ ]:

```
for j in num_neurons:
    clf = MLPClassifier(hidden_layer_sizes = (j))
    cv_results = cross_validate(clf, df_scaled, df.label.values, cv=3)
    print("Cross Validation Val Score for num_neurons = ", j, " is ",np.mean(cv_results[
'test_score']))
    train_acc = MLPClassifier(hidden_layer_sizes = (j)).fit(df_scaled, df.label.values).
score(df_scaled, df.label)
    print("Accuracy on the train set for num_neurons = ", j, " is ", train_acc)
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:5
71: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (200) reached and the op
timization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
/usr/local/lib/python3.7/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:5
71: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (200) reached and the op
timization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
/usr/local/lib/python3.7/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:5
71: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (200) reached and the op
timization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
```

```
Cross Validation Val Score for num_neurons =  4  is  0.9176029962546816
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:5
71: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (200) reached and the op
timization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
```

```
Accuracy on the train set for num_neurons =  4  is  0.9574468085106383
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:5
71: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (200) reached and the op
timization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
/usr/local/lib/python3.7/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:5
71: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (200) reached and the op
timization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
/usr/local/lib/python3.7/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:5
71: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (200) reached and the op
timization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
```

```
Cross Validation Val Score for num_neurons =  5  is  0.869674185463659
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:5
71: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (200) reached and the op
timization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
```

```
Accuracy on the train set for num_neurons =  5  is  0.9962453066332916
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:5
71: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (200) reached and the op
timization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
/usr/local/lib/python3.7/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:5
71: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (200) reached and the op
timization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
/usr/local/lib/python3.7/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:5
71: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (200) reached and the op
timization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
```

```
Cross Validation Val Score for num_neurons =  7  is  0.9135338345864662
Accuracy on the train set for num_neurons =  7  is  1.0
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:5
71: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (200) reached and the op
timization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
```

In [ ]:

```
best_model  = MLPClassifier(hidden_layer_sizes = (7)).fit(df.drop(columns="label"), df.l
abel)
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:5
71: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (200) reached and the op
timization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
```

In [ ]:

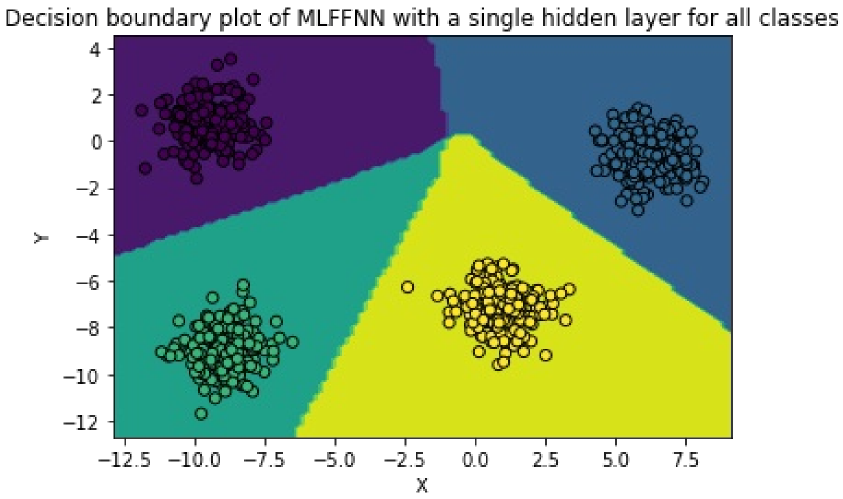
```
X = df[['x','y']].values
```

In [ ]:

```
xmin, xmax = np.min(X[:,0]) - 1, np.max(X[:,0]) + 1
ymin, ymax = np.min(X[:,1]) - 1, np.max(X[:,1]) + 1

x = np.linspace(xmin,xmax,100)
y = np.linspace(ymin,ymax,100)

xx, yy = np.meshgrid(x,y)

Z = np.stack((xx.ravel(),yy.ravel()),axis = 1)

pred = best_model.predict(Z).reshape(100,100)
# print(pred.shape)

plt.contourf(xx,yy,pred)
plt.scatter(X[:,0],X[:,1],c = df.label, edgecolors= 'k')
plt.xlabel("X")
plt.ylabel("Y")
plt.title("Decision boundary plot of MLFFNN with a single hidden layer for all classes")
```

Out[ ]:

```
Text(0.5, 1.0, 'Decision boundary plot of MLFFNN with a single hidden layer for all class
es')
```



In [ ]:

```
test_acc = best_model.score(df_test_scaled, df_test.label)
print("Accuracy on the test set is  ", test_acc)
```

```
Accuracy on the test set is   1.0
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:5
71: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (200) reached and the op
timization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
```

In [ ]:

```
confusion_matrix(df.label, best_model.predict(df.drop(columns= ["label"])))
```

Out[ ]:

```
array([[199,    0,    0,    0],
       [  0, 200,    0,    0]
       [   0, 200,    0,    0],
       [   0,    0, 200,    0],
       [   0,    0,    0, 200]])
```

In [ ]:

```
confusion_matrix(df_test.label, best_model.predict(df_test.drop(columns= ["label"])))
```

Out[ ]:

```
array([[30,  0,  0,  0],
       [ 0, 30,  0,  0],
       [ 0,  0, 29,  0],
       [ 0,  0,  0, 30]])
```

Linear SVM classifier for every pair of classes

0 v/s 1

In [ ]:

```
c = [0.0001, 0.001, 0.01, 0.1, 1, 10]
```

In [ ]:

```
gamma = [0.0001, 0.001, 0.01, 0.1, 1, 10]
```

In [ ]:

```
for i in c:
    for g in gamma:
        clf = SVC(C=i, gamma=g,  kernel='linear')
        cv_results = cross_validate(clf, df_0_1.drop(columns= ["label"]), df_0_1.label, 
cv=3)
        print("Cross Validation Test Score for C = ", i, "and gamma = ", g , "is",np.mea
n(cv_results['test_score']))
        clf = SVC(C=i, gamma=g,  kernel='linear')
        train_acc = clf.fit(df_0_1.drop(columns = ["label"]), df_0_1.label).score(df_0_1
.drop(columns = ["label"]), df_0_1.label)
        print("Train accuracy for C = ", i, "and gamma = ", g , "is",train_acc)
```

```
Cross Validation Test Score for C =  0.0001 and gamma =  0.0001 is 1.0
Train accuracy for C =  0.0001 and gamma =  0.0001 is 1.0
Cross Validation Test Score for C =  0.0001 and gamma =  0.001 is 1.0
Train accuracy for C =  0.0001 and gamma =  0.001 is 1.0
Cross Validation Test Score for C =  0.0001 and gamma =  0.01 is 1.0
Train accuracy for C =  0.0001 and gamma =  0.01 is 1.0
Cross Validation Test Score for C =  0.0001 and gamma =  0.1 is 1.0
Train accuracy for C =  0.0001 and gamma =  0.1 is 1.0
Cross Validation Test Score for C =  0.0001 and gamma =  1 is 1.0
Train accuracy for C =  0.0001 and gamma =  1 is 1.0
Cross Validation Test Score for C =  0.0001 and gamma =  10 is 1.0
Train accuracy for C =  0.0001 and gamma =  10 is 1.0
Cross Validation Test Score for C =  0.001 and gamma =  0.0001 is 1.0
Train accuracy for C =  0.001 and gamma =  0.0001 is 1.0
Cross Validation Test Score for C =  0.001 and gamma =  0.001 is 1.0
Train accuracy for C =  0.001 and gamma =  0.001 is 1.0
Cross Validation Test Score for C =  0.001 and gamma =  0.01 is 1.0
Train accuracy for C =  0.001 and gamma =  0.01 is 1.0
Cross Validation Test Score for C =  0.001 and gamma =  0.1 is 1.0
Train accuracy for C =  0.001 and gamma =  0.1 is 1.0
Cross Validation Test Score for C =  0.001 and gamma =  1 is 1.0
Train accuracy for C =  0.001 and gamma =  1 is 1.0
Cross Validation Test Score for C =  0.001 and gamma =  10 is 1.0
Train accuracy for C =  0.001 and gamma =  10 is 1.0
Cross Validation Test Score for C =  0.01 and gamma =  0.0001 is 1.0
Train accuracy for C =  0.01 and gamma =  0.0001 is 1.0
Cross Validation Test Score for C =  0.01 and gamma =  0.001 is 1.0
Train accuracy for C =  0.01 and gamma =  0.001 is 1.0
```

```
Cross Validation Test Score for C =  0.01 and gamma =  0.01 is 1.0
Train accuracy for C =  0.01 and gamma =  0.01 is 1.0
Cross Validation Test Score for C =  0.01 and gamma =  0.1 is 1.0
Train accuracy for C =  0.01 and gamma =  0.1 is 1.0
Cross Validation Test Score for C =  0.01 and gamma =  1 is 1.0
Train accuracy for C =  0.01 and gamma =  1 is 1.0
Cross Validation Test Score for C =  0.01 and gamma =  10 is 1.0
Train accuracy for C =  0.01 and gamma =  10 is 1.0
Cross Validation Test Score for C =  0.1 and gamma =  0.0001 is 1.0
Train accuracy for C =  0.1 and gamma =  0.0001 is 1.0
Cross Validation Test Score for C =  0.1 and gamma =  0.001 is 1.0
Train accuracy for C =  0.1 and gamma =  0.001 is 1.0
Cross Validation Test Score for C =  0.1 and gamma =  0.01 is 1.0
Train accuracy for C =  0.1 and gamma =  0.01 is 1.0
Cross Validation Test Score for C =  0.1 and gamma =  0.1 is 1.0
Train accuracy for C =  0.1 and gamma =  0.1 is 1.0
Cross Validation Test Score for C =  0.1 and gamma =  1 is 1.0
Train accuracy for C =  0.1 and gamma =  1 is 1.0
Cross Validation Test Score for C =  0.1 and gamma =  10 is 1.0
Train accuracy for C =  0.1 and gamma =  10 is 1.0
Cross Validation Test Score for C =  1 and gamma =  0.0001 is 1.0
Train accuracy for C =  1 and gamma =  0.0001 is 1.0
Cross Validation Test Score for C =  1 and gamma =  0.001 is 1.0
Train accuracy for C =  1 and gamma =  0.001 is 1.0
Cross Validation Test Score for C =  1 and gamma =  0.01 is 1.0
Train accuracy for C =  1 and gamma =  0.01 is 1.0
Cross Validation Test Score for C =  1 and gamma =  0.1 is 1.0
Train accuracy for C =  1 and gamma =  0.1 is 1.0
Cross Validation Test Score for C =  1 and gamma =  1 is 1.0
Train accuracy for C =  1 and gamma =  1 is 1.0
Cross Validation Test Score for C =  1 and gamma =  10 is 1.0
Train accuracy for C =  1 and gamma =  10 is 1.0
Cross Validation Test Score for C =  10 and gamma =  0.0001 is 1.0
Train accuracy for C =  10 and gamma =  0.0001 is 1.0
Cross Validation Test Score for C =  10 and gamma =  0.001 is 1.0
Train accuracy for C =  10 and gamma =  0.001 is 1.0
Cross Validation Test Score for C =  10 and gamma =  0.01 is 1.0
Train accuracy for C =  10 and gamma =  0.01 is 1.0
Cross Validation Test Score for C =  10 and gamma =  0.1 is 1.0
Train accuracy for C =  10 and gamma =  0.1 is 1.0
Cross Validation Test Score for C =  10 and gamma =  1 is 1.0
Train accuracy for C =  10 and gamma =  1 is 1.0
Cross Validation Test Score for C =  10 and gamma =  10 is 1.0
Train accuracy for C =  10 and gamma =  10 is 1.0
```

In [ ]:

```
clf = SVC(C=10, gamma=1,  kernel='linear')
test_acc = clf.fit(df_0_1.drop(columns = ["label"]), df_0_1.label).score(df_0_1_test.dro
p(columns = ["label"]), df_0_1_test.label)
print("Accuracy on the test set is : ", test_acc)
```

```
Accuracy on the test set is :  1.0
```

1 v/s 2

In [ ]:

```
for i in c:
    for g in gamma:
        clf = SVC(C=i, gamma=g,  kernel='linear')
        cv_results = cross_validate(clf, df_1_2.drop(columns= ["label"]), df_1_2.label,
cv=3)
        print("Cross Validation Test Score for C = ", i, "and gamma = ", g , "is",np.mea
n(cv_results['test_score']))
        clf = SVC(C=i, gamma=g,  kernel='linear')
        train_acc = clf.fit(df_1_2.drop(columns = ["label"]), df_1_2.label).score(df_1_2
.drop(columns = ["label"]), df_1_2.label)
        print("Train accuracy for C = ", i, "and gamma = ", g , "is",train_acc)
```

```
Cross Validation Test Score for C =  0.0001 and gamma =  0.0001 is 1.0
Cross Validation Test Score for C =  0.0001 and gamma =  0.0001 is 1.0
Cross Validation Test Score for C =  0.0001 and gamma =  0.001 is 1.0
Cross Validation Test Score for C =  0.0001 and gamma =  0.01 is 1.0
Cross Validation Test Score for C =  0.0001 and gamma =  0.1 is 1.0
Cross Validation Test Score for C =  0.0001 and gamma =  1 is 1.0
Cross Validation Test Score for C =  0.0001 and gamma =  10 is 1.0
Cross Validation Test Score for C =  0.001 and gamma =  0.0001 is 1.0
Cross Validation Test Score for C =  0.001 and gamma =  0.001 is 1.0
Cross Validation Test Score for C =  0.001 and gamma =  0.01 is 1.0
Cross Validation Test Score for C =  0.001 and gamma =  0.1 is 1.0
Cross Validation Test Score for C =  0.001 and gamma =  1 is 1.0
Cross Validation Test Score for C =  0.001 and gamma =  10 is 1.0
Cross Validation Test Score for C =  0.01 and gamma =  0.0001 is 1.0
Cross Validation Test Score for C =  0.01 and gamma =  0.001 is 1.0
Cross Validation Test Score for C =  0.01 and gamma =  0.01 is 1.0
Cross Validation Test Score for C =  0.01 and gamma =  0.1 is 1.0
Cross Validation Test Score for C =  0.01 and gamma =  1 is 1.0
Cross Validation Test Score for C =  0.01 and gamma =  10 is 1.0
Cross Validation Test Score for C =  0.1 and gamma =  0.0001 is 1.0
Cross Validation Test Score for C =  0.1 and gamma =  0.001 is 1.0
Cross Validation Test Score for C =  0.1 and gamma =  0.01 is 1.0
Cross Validation Test Score for C =  0.1 and gamma =  0.1 is 1.0
Cross Validation Test Score for C =  0.1 and gamma =  1 is 1.0
Cross Validation Test Score for C =  0.1 and gamma =  10 is 1.0
Cross Validation Test Score for C =  1 and gamma =  0.0001 is 1.0
Cross Validation Test Score for C =  1 and gamma =  0.001 is 1.0
Cross Validation Test Score for C =  1 and gamma =  0.01 is 1.0
Cross Validation Test Score for C =  1 and gamma =  0.1 is 1.0
Cross Validation Test Score for C =  1 and gamma =  1 is 1.0
Cross Validation Test Score for C =  1 and gamma =  10 is 1.0
Cross Validation Test Score for C =  10 and gamma =  0.0001 is 1.0
Cross Validation Test Score for C =  10 and gamma =  0.001 is 1.0
Cross Validation Test Score for C =  10 and gamma =  0.01 is 1.0
Cross Validation Test Score for C =  10 and gamma =  0.1 is 1.0
Cross Validation Test Score for C =  10 and gamma =  1 is 1.0
Cross Validation Test Score for C =  10 and gamma =  10 is 1.0
```

In [ ]:

```
clf = SVC(C=10, gamma=1,  kernel='linear')
test_acc = clf.fit(df_1_2.drop(columns = ["label"]), df_1_2.label).score(df_1_2_test.drop(columns = ["label"]), df_1_2_test.label)
print("Accuracy on the test set is : ", test_acc)
```

```
Accuracy on the test set is :  1.0
```

0 v/s 2

In [ ]:

```
for i in c:
    for g in gamma:
        clf = SVC(C=i, gamma=g,  kernel='linear')
        cv_results = cross_validate(clf, df_0_2.drop(columns= ["label"]), df_0_2.label, cv=3)
        print("Cross Validation Test Score for C = ", i, "and gamma = ", g , "is",np.mean(cv_results['test_score']))
```

```
Cross Validation Test Score for C =  0.0001 and gamma =  0.0001 is 1.0
Cross Validation Test Score for C =  0.0001 and gamma =  0.001 is 1.0
Cross Validation Test Score for C =  0.0001 and gamma =  0.01 is 1.0
Cross Validation Test Score for C =  0.0001 and gamma =  0.1 is 1.0
Cross Validation Test Score for C =  0.0001 and gamma =  1 is 1.0
Cross Validation Test Score for C =  0.0001 and gamma =  10 is 1.0
Cross Validation Test Score for C =  0.001 and gamma =  0.0001 is 1.0
Cross Validation Test Score for C =  0.001 and gamma =  0.001 is 1.0
Cross Validation Test Score for C =  0.001 and gamma =  0.01 is 1.0
Cross Validation Test Score for C =  0.001 and gamma =  0.1 is 1.0
Cross Validation Test Score for C =  0.001 and gamma =  1 is 1.0
Cross Validation Test Score for C =  0.001 and gamma =  10 is 1.0
Cross Validation Test Score for C =  0.01 and gamma =  0.0001 is 1.0
Cross Validation Test Score for C =  0.01 and gamma =  0.001 is 1.0
Cross Validation Test Score for C =  0.01 and gamma =  0.01 is 1.0
Cross Validation Test Score for C =  0.01 and gamma =  0.1 is 1.0
Cross Validation Test Score for C =  0.01 and gamma =  1 is 1.0
Cross Validation Test Score for C =  0.01 and gamma =  10 is 1.0
Cross Validation Test Score for C =  0.1 and gamma =  0.0001 is 1.0
Cross Validation Test Score for C =  0.1 and gamma =  0.001 is 1.0
Cross Validation Test Score for C =  0.1 and gamma =  0.01 is 1.0
Cross Validation Test Score for C =  0.1 and gamma =  0.1 is 1.0
Cross Validation Test Score for C =  0.1 and gamma =  1 is 1.0
Cross Validation Test Score for C =  0.1 and gamma =  10 is 1.0
Cross Validation Test Score for C =  1 and gamma =  0.0001 is 1.0
Cross Validation Test Score for C =  1 and gamma =  0.001 is 1.0
Cross Validation Test Score for C =  1 and gamma =  0.01 is 1.0
Cross Validation Test Score for C =  1 and gamma =  0.1 is 1.0
Cross Validation Test Score for C =  1 and gamma =  1 is 1.0
Cross Validation Test Score for C =  1 and gamma =  10 is 1.0
Cross Validation Test Score for C =  10 and gamma =  0.0001 is 1.0
Cross Validation Test Score for C =  10 and gamma =  0.001 is 1.0
Cross Validation Test Score for C =  10 and gamma =  0.01 is 1.0
Cross Validation Test Score for C =  10 and gamma =  0.1 is 1.0
Cross Validation Test Score for C =  10 and gamma =  1 is 1.0
Cross Validation Test Score for C =  10 and gamma =  10 is 1.0
```

In [ ]:

```python
clf = SVC(C=10, gamma=1,  kernel='linear')
test_acc = clf.fit(df_0_2.drop(columns = ["label"]), df_0_2.label).score(df_0_2_test.dro
p(columns = ["label"]), df_0_2_test.label)
print("Accuracy on the test set is : ", test_acc)
```

```
Accuracy on the test set is :  1.0
```

0 v/s 3

In [ ]:

```python
for i in c:
    for g in gamma:
        clf = SVC(C=i, gamma=g,  kernel='linear')
        cv_results = cross_validate(clf, df_0_3.drop(columns= ["label"]), df_0_3.label,
cv=3)
        print("Cross Validation Test Score for C = ", i, "and gamma = ", g , "is",np.mea
n(cv_results['test_score']))
        clf = SVC(C=i, gamma=g,  kernel='linear')
        train_acc = clf.fit(df_0_3.drop(columns = ["label"]), df_0_3.label).score(df_0_3
.drop(columns = ["label"]), df_0_3.label)
        print("Train accuracy for C = ", i, "and gamma = ", g , "is",train_acc)
```

```
Cross Validation Test Score for C =  0.0001 and gamma =  0.0001 is 1.0
Train accuracy for C =  0.0001 and gamma =  0.0001 is 1.0
Cross Validation Test Score for C =  0.0001 and gamma =  0.001 is 1.0
Train accuracy for C =  0.0001 and gamma =  0.001 is 1.0
Cross Validation Test Score for C =  0.0001 and gamma =  0.01 is 1.0
Train accuracy for C =  0.0001 and gamma =  0.01 is 1.0
Cross Validation Test Score for C =  0.0001 and gamma =  0.1 is 1.0
Train accuracy for C =  0.0001 and gamma =  0.1 is 1.0
Cross Validation Test Score for C =  0.0001 and gamma =  1 is 1.0
Train accuracy for C =  0.0001 and gamma =  1 is 1.0
Cross Validation Test Score for C =  0.001 and gamma =  0.0001 is 1.0
Train accuracy for C =  0.001 and gamma =  0.0001 is 1.0
Cross Validation Test Score for C =  0.001 and gamma =  0.001 is 1.0
Train accuracy for C =  0.001 and gamma =  0.001 is 1.0
Cross Validation Test Score for C =  0.001 and gamma =  0.01 is 1.0
Train accuracy for C =  0.001 and gamma =  0.01 is 1.0
Cross Validation Test Score for C =  0.001 and gamma =  0.1 is 1.0
Train accuracy for C =  0.001 and gamma =  0.1 is 1.0
Cross Validation Test Score for C =  0.001 and gamma =  1 is 1.0
Train accuracy for C =  0.001 and gamma =  1 is 1.0
Cross Validation Test Score for C =  0.01 and gamma =  0.0001 is 1.0
Train accuracy for C =  0.01 and gamma =  0.0001 is 1.0
Train accuracy for C =  0.01 and gamma =  0.0001 is 1.0
Cross Validation Test Score for C =  0.01 and gamma =  0.001 is 1.0
Train accuracy for C =  0.01 and gamma =  0.001 is 1.0
Cross Validation Test Score for C =  0.01 and gamma =  0.01 is 1.0
Train accuracy for C =  0.01 and gamma =  0.01 is 1.0
Cross Validation Test Score for C =  0.01 and gamma =  0.1 is 1.0
Train accuracy for C =  0.01 and gamma =  0.1 is 1.0
Cross Validation Test Score for C =  0.01 and gamma =  1 is 1.0
Train accuracy for C =  0.01 and gamma =  1 is 1.0
Cross Validation Test Score for C =  0.1 and gamma =  0.0001 is 1.0
Train accuracy for C =  0.1 and gamma =  0.0001 is 1.0
Cross Validation Test Score for C =  0.1 and gamma =  0.001 is 1.0
Train accuracy for C =  0.1 and gamma =  0.001 is 1.0
Cross Validation Test Score for C =  0.1 and gamma =  0.01 is 1.0
Train accuracy for C =  0.1 and gamma =  0.01 is 1.0
Cross Validation Test Score for C =  0.1 and gamma =  0.1 is 1.0
Train accuracy for C =  0.1 and gamma =  0.1 is 1.0
Cross Validation Test Score for C =  0.1 and gamma =  1 is 1.0
Train accuracy for C =  0.1 and gamma =  1 is 1.0
Cross Validation Test Score for C =  10 and gamma =  0.0001 is 1.0
Train accuracy for C =  10 and gamma =  0.0001 is 1.0
Cross Validation Test Score for C =  10 and gamma =  0.001 is 1.0
Train accuracy for C =  10 and gamma =  0.001 is 1.0
Cross Validation Test Score for C =  10 and gamma =  0.01 is 1.0
Train accuracy for C =  10 and gamma =  0.01 is 1.0
Cross Validation Test Score for C =  10 and gamma =  0.1 is 1.0
Train accuracy for C =  10 and gamma =  0.1 is 1.0
Cross Validation Test Score for C =  10 and gamma =  1 is 1.0
Train accuracy for C =  10 and gamma =  1 is 1.0
```

In [ ]:

```
clf = SVC(C=10, gamma=1,  kernel='linear')
test_acc = clf.fit(df_0_3.drop(columns = ["label"]), df_0_3.label).score(df_0_3_test.drop(columns = ["label"]), df_0_3_test.label)
print("Accuracy on the test set is : ", test_acc)
```

```
Accuracy on the test set is :  1.0
```

1 v/s 3

In [ ]:

```
for i in c:
    for g in gamma:
        clf = SVC(C=i, gamma=g,  kernel='linear')
        cv_results = cross_validate(clf, df_1_3.drop(columns= ["label"]), df_1_3.label, cv=3)
        print("Cross Validation Test Score for C = ", i, "and gamma = ", g , "is",np.mean(cv_results['test_score']))
        clf = SVC(C=i, gamma=g,  kernel='linear')
        train_acc = clf.fit(df_1_3.drop(columns = ["label"]), df_1_3.label).score(df_1_3.drop(columns = ["label"]), df_1_3.label)
        print("Train accuracy for C = ", i, "and gamma = ", g , "is",train_acc)
```

```
Cross Validation Test Score for C =  0.0001 and gamma =  0.0001 is 0.9874686716791979
Train accuracy for C =  0.0001 and gamma =  0.0001 is 1.0
Cross Validation Test Score for C =  0.0001 and gamma =  0.001 is 0.9874686716791979
Train accuracy for C =  0.0001 and gamma =  0.001 is 1.0
Cross Validation Test Score for C =  0.0001 and gamma =  0.01 is 0.9874686716791979
Train accuracy for C =  0.0001 and gamma =  0.01 is 1.0
Cross Validation Test Score for C =  0.0001 and gamma =  0.1 is 0.9874686716791979
Train accuracy for C =  0.0001 and gamma =  0.1 is 1.0
Cross Validation Test Score for C =  0.0001 and gamma =  1 is 0.9874686716791979
Train accuracy for C =  0.0001 and gamma =  1 is 1.0
Cross Validation Test Score for C =  0.001 and gamma =  0.0001 is 1.0
Train accuracy for C =  0.001 and gamma =  0.0001 is 1.0
Cross Validation Test Score for C =  0.001 and gamma =  0.001 is 1.0
Train accuracy for C =  0.001 and gamma =  0.001 is 1.0
Cross Validation Test Score for C =  0.001 and gamma =  0.01 is 1.0
Train accuracy for C =  0.001 and gamma =  0.01 is 1.0
Train accuracy for C =  0.001 and gamma =  0.01 is 1.0
Cross Validation Test Score for C =  0.001 and gamma =  0.1 is 1.0
Train accuracy for C =  0.001 and gamma =  0.1 is 1.0
Cross Validation Test Score for C =  0.001 and gamma =  1 is 1.0
Train accuracy for C =  0.001 and gamma =  1 is 1.0
Cross Validation Test Score for C =  0.01 and gamma =  0.0001 is 1.0
Train accuracy for C =  0.01 and gamma =  0.0001 is 1.0
Cross Validation Test Score for C =  0.01 and gamma =  0.001 is 1.0
Train accuracy for C =  0.01 and gamma =  0.001 is 1.0
Cross Validation Test Score for C =  0.01 and gamma =  0.01 is 1.0
Train accuracy for C =  0.01 and gamma =  0.01 is 1.0
Cross Validation Test Score for C =  0.01 and gamma =  0.1 is 1.0
Train accuracy for C =  0.01 and gamma =  0.1 is 1.0
Cross Validation Test Score for C =  0.01 and gamma =  1 is 1.0
Train accuracy for C =  0.01 and gamma =  1 is 1.0
Cross Validation Test Score for C =  0.1 and gamma =  0.0001 is 1.0
Train accuracy for C =  0.1 and gamma =  0.0001 is 1.0
Cross Validation Test Score for C =  0.1 and gamma =  0.001 is 1.0
Train accuracy for C =  0.1 and gamma =  0.001 is 1.0
Cross Validation Test Score for C =  0.1 and gamma =  0.01 is 1.0
Train accuracy for C =  0.1 and gamma =  0.01 is 1.0
Cross Validation Test Score for C =  0.1 and gamma =  0.1 is 1.0
Train accuracy for C =  0.1 and gamma =  0.1 is 1.0
Cross Validation Test Score for C =  0.1 and gamma =  1 is 1.0
Train accuracy for C =  0.1 and gamma =  1 is 1.0
Cross Validation Test Score for C =  10 and gamma =  0.0001 is 1.0
Train accuracy for C =  10 and gamma =  0.0001 is 1.0
Cross Validation Test Score for C =  10 and gamma =  0.001 is 1.0
Train accuracy for C =  10 and gamma =  0.001 is 1.0
Cross Validation Test Score for C =  10 and gamma =  0.01 is 1.0
Train accuracy for C =  10 and gamma =  0.01 is 1.0
Cross Validation Test Score for C =  10 and gamma =  0.1 is 1.0
Train accuracy for C =  10 and gamma =  0.1 is 1.0
Cross Validation Test Score for C =  10 and gamma =  1 is 1.0
Train accuracy for C =  10 and gamma =  1 is 1.0
```

In [ ]:

```
clf = SVC(C=10, gamma=1,  kernel='linear')
test_acc = clf.fit(df_1_3.drop(columns = ["label"]), df_1_3.label).score(df_1_3_test.dro
p(columns = ["label"]), df_1_3_test.label)
print("Accuracy on the test set is : ", test_acc)
```

```
Accuracy on the test set is :  1.0
```

2 v/s 3

In [ ]:

```
for i in c:
    for g in gamma:
        clf = SVC(C=i, gamma=g,  kernel='linear')
        cv_results = cross_validate(clf, df_2_3.drop(columns= ["label"]), df_2_3.label,
cv=3)
        print("Cross Validation Test Score for C = ", i, "and gamma = ", g , "is",np.mea
n(cv_results['test_score']))
        clf = SVC(C=i, gamma=g,  kernel='linear')
        train_acc = clf.fit(df_2_3.drop(columns = ["label"]), df_2_3.label).score(df_2_3
.drop(columns = ["label"]), df_2_3.label)
        print("Train accuracy for C = ", i, "and gamma = ", g , "is",train_acc)
```

```
Cross Validation Test Score for C =  0.0001 and gamma =  0.0001 is 1.0
Train accuracy for C =  0.0001 and gamma =  0.0001 is 1.0
Cross Validation Test Score for C =  0.0001 and gamma =  0.001 is 1.0
Train accuracy for C =  0.0001 and gamma =  0.001 is 1.0
Cross Validation Test Score for C =  0.0001 and gamma =  0.01 is 1.0
Train accuracy for C =  0.0001 and gamma =  0.01 is 1.0
Cross Validation Test Score for C =  0.0001 and gamma =  0.1 is 1.0
Train accuracy for C =  0.0001 and gamma =  0.1 is 1.0
Cross Validation Test Score for C =  0.0001 and gamma =  1 is 1.0
Train accuracy for C =  0.0001 and gamma =  1 is 1.0
```

```
Cross Validation Test Score for C =  0.001 and gamma =  0.0001 is 1.0
Train accuracy for C =  0.001 and gamma =  0.0001 is 1.0
Cross Validation Test Score for C =  0.001 and gamma =  0.001 is 1.0
Train accuracy for C =  0.001 and gamma =  0.001 is 1.0
Cross Validation Test Score for C =  0.001 and gamma =  0.01 is 1.0
Train accuracy for C =  0.001 and gamma =  0.01 is 1.0
Cross Validation Test Score for C =  0.001 and gamma =  0.1 is 1.0
Train accuracy for C =  0.001 and gamma =  0.1 is 1.0
Cross Validation Test Score for C =  0.001 and gamma =  1 is 1.0
Train accuracy for C =  0.001 and gamma =  1 is 1.0
Cross Validation Test Score for C =  0.01 and gamma =  0.0001 is 1.0
Train accuracy for C =  0.01 and gamma =  0.0001 is 1.0
Cross Validation Test Score for C =  0.01 and gamma =  0.001 is 1.0
Train accuracy for C =  0.01 and gamma =  0.001 is 1.0
Cross Validation Test Score for C =  0.01 and gamma =  0.01 is 1.0
Train accuracy for C =  0.01 and gamma =  0.01 is 1.0
Cross Validation Test Score for C =  0.01 and gamma =  0.1 is 1.0
Train accuracy for C =  0.01 and gamma =  0.1 is 1.0
Cross Validation Test Score for C =  0.01 and gamma =  1 is 1.0
Train accuracy for C =  0.01 and gamma =  1 is 1.0
Cross Validation Test Score for C =  0.1 and gamma =  0.0001 is 1.0
Train accuracy for C =  0.1 and gamma =  0.0001 is 1.0
Cross Validation Test Score for C =  0.1 and gamma =  0.001 is 1.0
Train accuracy for C =  0.1 and gamma =  0.001 is 1.0
Cross Validation Test Score for C =  0.1 and gamma =  0.01 is 1.0
Train accuracy for C =  0.1 and gamma =  0.01 is 1.0
Cross Validation Test Score for C =  0.1 and gamma =  0.1 is 1.0
Train accuracy for C =  0.1 and gamma =  0.1 is 1.0
Cross Validation Test Score for C =  0.1 and gamma =  1 is 1.0
Train accuracy for C =  0.1 and gamma =  1 is 1.0
Cross Validation Test Score for C =  10 and gamma =  0.0001 is 1.0
Train accuracy for C =  10 and gamma =  0.0001 is 1.0
Cross Validation Test Score for C =  10 and gamma =  0.001 is 1.0
Train accuracy for C =  10 and gamma =  0.001 is 1.0
Cross Validation Test Score for C =  10 and gamma =  0.01 is 1.0
Train accuracy for C =  10 and gamma =  0.01 is 1.0
Cross Validation Test Score for C =  10 and gamma =  0.1 is 1.0
Train accuracy for C =  10 and gamma =  0.1 is 1.0
Cross Validation Test Score for C =  10 and gamma =  1 is 1.0
Train accuracy for C =  10 and gamma =  1 is 1.0
```

In [ ]:

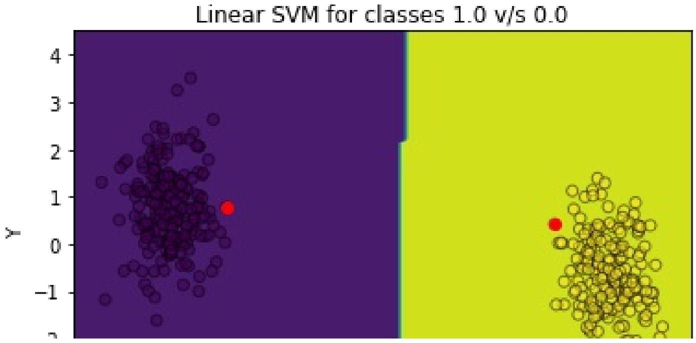
```
clf = SVC(C=10, gamma=1,  kernel='linear')
test_acc = clf.fit(df_2_3.drop(columns = ["label"]), df_2_3.label).score(df_2_3_test.drop(columns = ["label"]), df_2_3_test.label)
print("Accuracy on the test set is : ", test_acc)
```

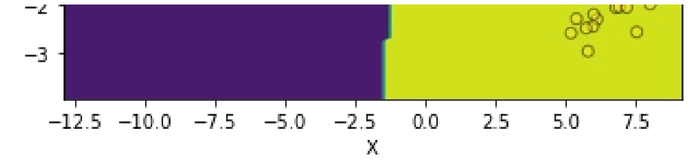
```
Accuracy on the test set is :  1.0
```

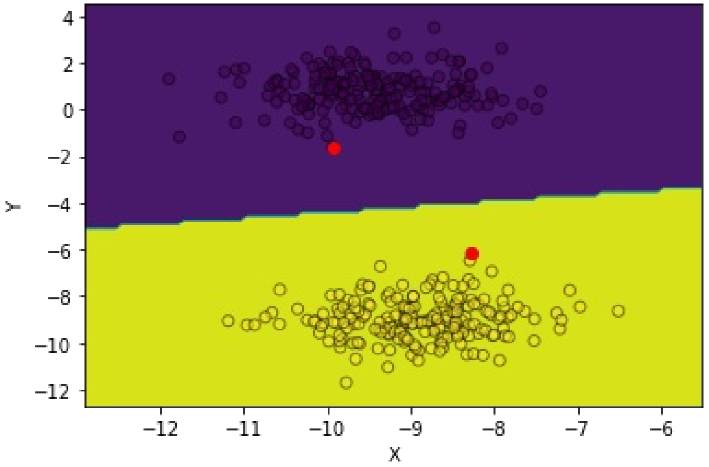
Plotting decision boundaries for pairwise linear SVMs

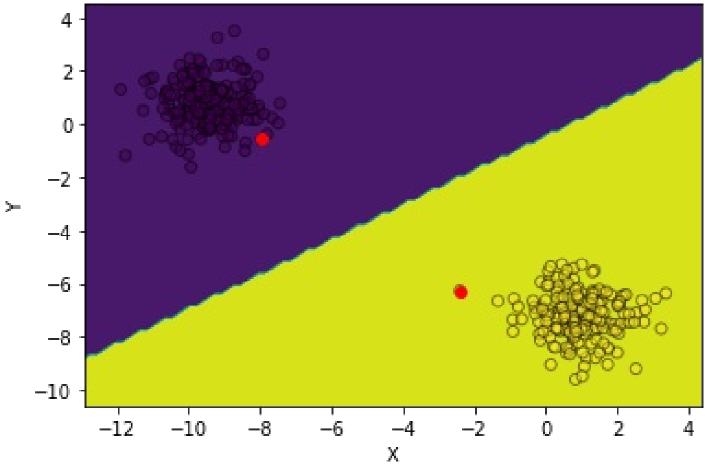
In [ ]:

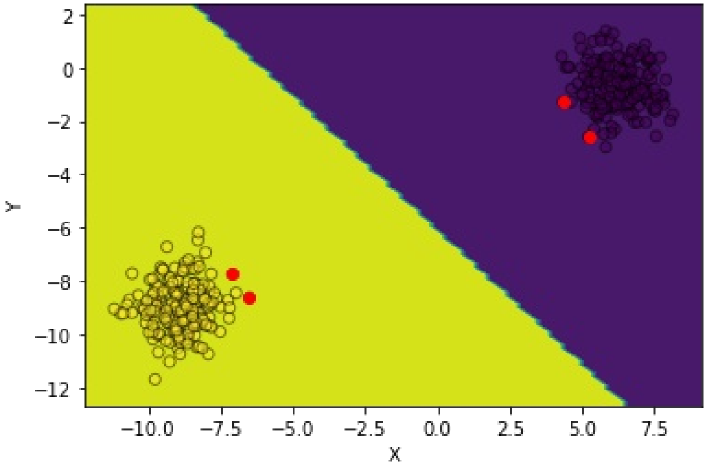
```
for df in [df_0_1, df_0_2, df_0_3, df_1_2, df_1_3, df_2_3]:
    best_model = SVC(C=10, gamma=1,  kernel='linear').fit(df.drop(columns= ["label"]), df.label)
    plot_pairs_svm_no_scaling(best_model, df)
```

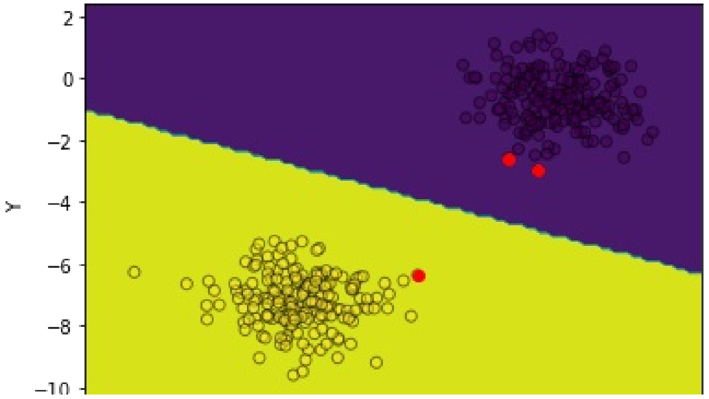


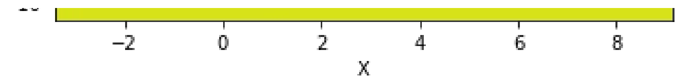


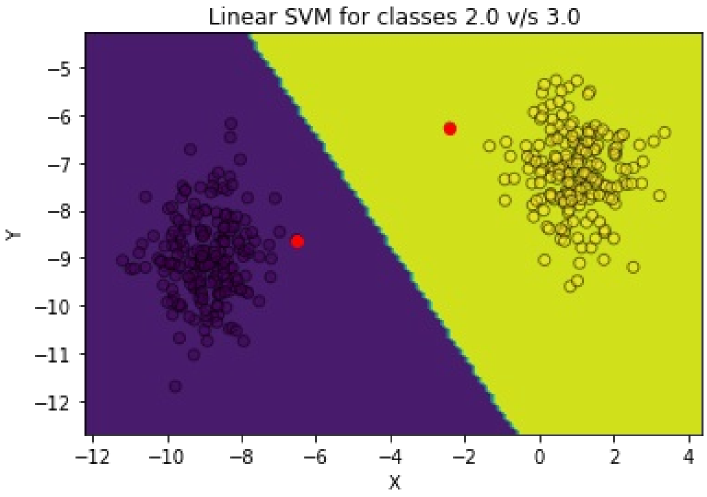

Linear SVM for classes 2.0 v/s 0.0


Linear SVM for classes 3.0 v/s 0.0


Linear SVM for classes 2.0 v/s 1.0


Linear SVM for classes 1.0 v/s 3.0





Dataset 1B

In [ ]:

```
df2 = pd.read_csv("/content/train (1).csv")
```

In [ ]:

```
df2.columns = ["x", "y", "label"]
```

In [ ]:

```
df2.head()
```

Out[ ]:

	x	y	label
0	0.963906	-0.084686	0.0
1	0.751577	-0.692578	0.0
2	0.064676	-1.029066	0.0
3	0.945798	-0.525876	0.0
4	1.057655	0.000716	0.0

In [ ]:

```
df2_test = pd.read_csv("/content/dev (1).csv")
```

In [ ]:

```
df2_test.columns = ["x", "y", "label"]
```

In [ ]:

```
df2_test.head()
```

Out[ ]:

	x	y	label
0	0.355078	-0.901234	0.0
1	-0.188768	-0.770182	0.0
2	0.428097	-0.988325	0.0
3	0.397620	-0.977080	0.0

	x	y	label
3	0.397020	-0.977080	0.0
4	-0.332474	-1.060686	0.0

In [ ]:

```
scaler = StandardScaler()
scaler.fit(df2.drop(columns = ["label"]))
df2_scaled = scaler.transform(df2.drop(columns  = ["label"]))
df2_test_scaled = scaler.transform(df2_test.drop(columns  = ["label"]))
```

MLP

In [ ]:

```
learning_rate = [0.00001, 0.0001, 0.001, 0.01, 0.1]
```

In [ ]:

```
hidden_layer_size_1 = [2,3]
hidden_layer_size_2 = [1,2,3]
```

In [ ]:

```
for i in learning_rate:
    for j in hidden_layer_size_1:
        for k in hidden_layer_size_2:
            clf = MLPClassifier(solver='lbfgs', alpha=i,hidden_layer_sizes=(j, k), rando
m_state=42)
            cv_results = cross_validate(clf, df2_scaled, df2.label, cv=3)
            print("Cross Validation Test Score for learning rate = ", i, "and number of
hidden units layer 1 = ", j ,"and number of hidden units layer 2 = ", k , "is",np.mean(c
v_results['test_score']))
            clf = MLPClassifier(solver='lbfgs', alpha=i,hidden_layer_sizes=(j, k), rando
m_state=42)
            train_acc = clf.fit(df2_scaled, df2.label).score(df2_scaled, df2.label)
            print("Train accuracy for learning rate = ", i, "and number of hidden units
layer 1 = ", j ,"and number of hidden units layer 2 = ", k , "is", train_acc)
```

```
Cross Validation Test Score for learning rate =  1e-05 and number of hidden units layer 1
=  2 and number of hidden units layer 2 =  1 is 0.7477638190954773
Train accuracy for learning rate =  1e-05 and number of hidden units layer 1 =  2 and num
ber of hidden units layer 2 =  1 is 0.8831385642737897
Cross Validation Test Score for learning rate =  1e-05 and number of hidden units layer 1
=  2 and number of hidden units layer 2 =  2 is 0.9430653266331658
Train accuracy for learning rate =  1e-05 and number of hidden units layer 1 =  2 and num
ber of hidden units layer 2 =  2 is 0.333889816360601
Cross Validation Test Score for learning rate =  1e-05 and number of hidden units layer 1
=  2 and number of hidden units layer 2 =  3 is 0.9683165829145729
Train accuracy for learning rate =  1e-05 and number of hidden units layer 1 =  2 and num
ber of hidden units layer 2 =  3 is 0.9449081803005008
Cross Validation Test Score for learning rate =  1e-05 and number of hidden units layer 1
=  3 and number of hidden units layer 2 =  1 is 0.7763735343383584
Train accuracy for learning rate =  1e-05 and number of hidden units layer 1 =  3 and num
ber of hidden units layer 2 =  1 is 0.8213689482470785
Cross Validation Test Score for learning rate =  1e-05 and number of hidden units layer 1
=  3 and number of hidden units layer 2 =  2 is 0.33221943048576213
Train accuracy for learning rate =  1e-05 and number of hidden units layer 1 =  3 and num
ber of hidden units layer 2 =  2 is 0.333889816360601
Cross Validation Test Score for learning rate =  1e-05 and number of hidden units layer 1
=  3 and number of hidden units layer 2 =  3 is 0.9966499162479062
Train accuracy for learning rate =  1e-05 and number of hidden units layer 1 =  3 and num
ber of hidden units layer 2 =  3 is 0.9766277128547579
Cross Validation Test Score for learning rate =  0.0001 and number of hidden units layer
1 =  2 and number of hidden units layer 2 =  1 is 0.7527973199329984
Train accuracy for learning rate =  0.0001 and number of hidden units layer 1 =  2 and nu
mber of hidden units layer 2 =  1 is 0.8781302170283807
Cross Validation Test Score for learning rate =  0.0001 and number of hidden units layer
1 =  2 and number of hidden units layer 2 =  2 is 0.9480904522613066
Train accuracy for learning rate =  0.0001 and number of hidden units layer 1 =  2 and nu
mber of hidden units layer 2 =  2 is 0.333889816360601
mber of hidden units layer 2 =  2 is 0.333889816360601
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:4
70: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
  self.n_iter_ = _check_optimize_result("lbfgs", opt_res, self.max_iter)
```

```
Cross Validation Test Score for learning rate =  0.0001 and number of hidden units layer
1 =  2 and number of hidden units layer 2 =  3 is 0.9599916247906197
Train accuracy for learning rate =  0.0001 and number of hidden units layer 1 =  2 and nu
mber of hidden units layer 2 =  3 is 0.9181969949916527
Cross Validation Test Score for learning rate =  0.0001 and number of hidden units layer
1 =  3 and number of hidden units layer 2 =  1 is 0.7713484087102177
Train accuracy for learning rate =  0.0001 and number of hidden units layer 1 =  3 and nu
mber of hidden units layer 2 =  1 is 0.8213689482470785
Cross Validation Test Score for learning rate =  0.0001 and number of hidden units layer
1 =  3 and number of hidden units layer 2 =  2 is 0.33221943048576213
Train accuracy for learning rate =  0.0001 and number of hidden units layer 1 =  3 and nu
mber of hidden units layer 2 =  2 is 0.333889816360601
Cross Validation Test Score for learning rate =  0.0001 and number of hidden units layer
1 =  3 and number of hidden units layer 2 =  3 is 0.9330150753768844
Train accuracy for learning rate =  0.0001 and number of hidden units layer 1 =  3 and nu
mber of hidden units layer 2 =  3 is 0.9582637729549248
Cross Validation Test Score for learning rate =  0.001 and number of hidden units layer 1
=  2 and number of hidden units layer 2 =  1 is 0.7963065326633165
Train accuracy for learning rate =  0.001 and number of hidden units layer 1 =  2 and num
ber of hidden units layer 2 =  1 is 0.8297161936560935
Cross Validation Test Score for learning rate =  0.001 and number of hidden units layer 1
=  2 and number of hidden units layer 2 =  2 is 0.9463986599664992
Train accuracy for learning rate =  0.001 and number of hidden units layer 1 =  2 and num
ber of hidden units layer 2 =  2 is 0.333889816360601
Cross Validation Test Score for learning rate =  0.001 and number of hidden units layer 1
=  2 and number of hidden units layer 2 =  3 is 0.9049497487437187
Train accuracy for learning rate =  0.001 and number of hidden units layer 1 =  2 and num
ber of hidden units layer 2 =  3 is 1.0
Cross Validation Test Score for learning rate =  0.001 and number of hidden units layer 1
=  3 and number of hidden units layer 2 =  1 is 0.7747068676716918
Train accuracy for learning rate =  0.001 and number of hidden units layer 1 =  3 and num
ber of hidden units layer 2 =  1 is 0.8213689482470785
Cross Validation Test Score for learning rate =  0.001 and number of hidden units layer 1
=  3 and number of hidden units layer 2 =  2 is 0.33221943048576213
Train accuracy for learning rate =  0.001 and number of hidden units layer 1 =  3 and num
ber of hidden units layer 2 =  2 is 0.333889816360601
Cross Validation Test Score for learning rate =  0.001 and number of hidden units layer 1
=  3 and number of hidden units layer 2 =  3 is 0.9916331658291457
Train accuracy for learning rate =  0.001 and number of hidden units layer 1 =  3 and num
ber of hidden units layer 2 =  3 is 0.7929883138564274
Cross Validation Test Score for learning rate =  0.01 and number of hidden units layer 1
=  2 and number of hidden units layer 2 =  1 is 0.7411809045226131
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:4
70: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
  self.n_iter_ = _check_optimize_result("lbfgs", opt_res, self.max_iter)
```

```
Train accuracy for learning rate =  0.01 and number of hidden units layer 1 =  2 and numb
er of hidden units layer 2 =  1 is 0.8831385642737897
Cross Validation Test Score for learning rate =  0.01 and number of hidden units layer 1
=  2 and number of hidden units layer 2 =  2 is 0.9112227805695142
Train accuracy for learning rate =  0.01 and number of hidden units layer 1 =  2 and numb
er of hidden units layer 2 =  2 is 0.333889816360601
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:4
70: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
    https://scikit-learn.org/stable/modules/preprocessing.html
  self.n_iter_ = _check_optimize_result("lbfgs", opt_res, self.max_iter)
/usr/local/lib/python3.7/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:4
70: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
  self.n_iter_ = _check_optimize_result("lbfgs", opt_res, self.max_iter)
/usr/local/lib/python3.7/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:4
70: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
  self.n_iter_ = _check_optimize_result("lbfgs", opt_res, self.max_iter)
```

```
Cross Validation Test Score for learning rate =  0.01 and number of hidden units layer 1
=  2 and number of hidden units layer 2 =  3 is 0.9416331658291458
Train accuracy for learning rate =  0.01 and number of hidden units layer 1 =  2 and numb
er of hidden units layer 2 =  3 is 0.8681135225375626
Cross Validation Test Score for learning rate =  0.01 and number of hidden units layer 1
=  3 and number of hidden units layer 2 =  1 is 0.7863735343383585
Train accuracy for learning rate =  0.01 and number of hidden units layer 1 =  3 and numb
er of hidden units layer 2 =  1 is 0.7278797996661102
Cross Validation Test Score for learning rate =  0.01 and number of hidden units layer 1
=  3 and number of hidden units layer 2 =  2 is 0.33221943048576213
Train accuracy for learning rate =  0.01 and number of hidden units layer 1 =  3 and numb
er of hidden units layer 2 =  2 is 0.333889816360601
Cross Validation Test Score for learning rate =  0.01 and number of hidden units layer 1
=  3 and number of hidden units layer 2 =  3 is 0.9533333333333333
Train accuracy for learning rate =  0.01 and number of hidden units layer 1 =  3 and numb
er of hidden units layer 2 =  3 is 0.9766277128547579
Cross Validation Test Score for learning rate =  0.1 and number of hidden units layer 1 =
2 and number of hidden units layer 2 =  1 is 0.707998324958124
Train accuracy for learning rate =  0.1 and number of hidden units layer 1 =  2 and numbe
r of hidden units layer 2 =  1 is 0.7011686143572621
Cross Validation Test Score for learning rate =  0.1 and number of hidden units layer 1 =
2 and number of hidden units layer 2 =  2 is 0.9547738693467336
Train accuracy for learning rate =  0.1 and number of hidden units layer 1 =  2 and numbe
r of hidden units layer 2 =  2 is 0.333889816360601
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:4
70: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
  self.n_iter_ = _check_optimize_result("lbfgs", opt_res, self.max_iter)
/usr/local/lib/python3.7/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:4
70: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
  self.n_iter_ = _check_optimize_result("lbfgs", opt_res, self.max_iter)
/usr/local/lib/python3.7/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:4
70: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
  self.n_iter_ = _check_optimize_result("lbfgs", opt_res, self.max_iter)
```

```
Cross Validation Test Score for learning rate =  0.1 and number of hidden units layer 1 =
2 and number of hidden units layer 2 =  3 is 0.9983249581239532
Train accuracy for learning rate =  0.1 and number of hidden units layer 1 =  2 and numbe
r of hidden units layer 2 =  3 is 0.9766277128547579
Cross Validation Test Score for learning rate =  0.1 and number of hidden units layer 1 =
3 and number of hidden units layer 2 =  1 is 0.7628475711892797
Train accuracy for learning rate =  0.1 and number of hidden units layer 1 =  3 and numbe
r of hidden units layer 2 =  1 is 0.8030050083472454
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:4
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:4
70: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
  self.n_iter_ = _check_optimize_result("lbfgs", opt_res, self.max_iter)
```

```
Cross Validation Test Score for learning rate =  0.1 and number of hidden units layer 1 = 
3 and number of hidden units layer 2 =  2 is 0.3305527638190955
Train accuracy for learning rate =  0.1 and number of hidden units layer 1 =  3 and numbe
r of hidden units layer 2 =  2 is 0.333889816360601
Cross Validation Test Score for learning rate =  0.1 and number of hidden units layer 1 = 
3 and number of hidden units layer 2 =  3 is 0.9449916247906197
Train accuracy for learning rate =  0.1 and number of hidden units layer 1 =  3 and numbe
r of hidden units layer 2 =  3 is 1.0
```

In [ ]:

```
best_model  = MLPClassifier(solver='lbfgs', alpha=0.1,hidden_layer_sizes=(3, 3), random_
state=42).fit(df2_scaled, df2.label)
```

In [ ]:

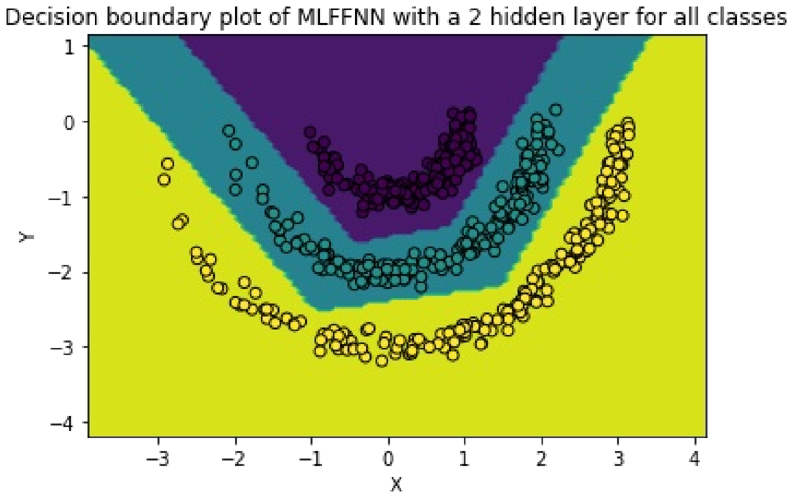
```
X = df2[['x','y']].values
```

In [ ]:

```
xmin, xmax = np.min(X[:,0]) - 1, np.max(X[:,0]) + 1
ymin, ymax = np.min(X[:,1]) - 1, np.max(X[:,1]) + 1

x = np.linspace(xmin,xmax,100)
y = np.linspace(ymin,ymax,100)

xx, yy = np.meshgrid(x,y)

Z = np.stack((xx.ravel(),yy.ravel()),axis = 1)

Z = scaler.transform(Z)
pred = best_model.predict(Z).reshape(100,100)
# print(pred.shape)

plt.contourf(xx,yy,pred)
plt.scatter(X[:,0],X[:,1],c = df2.label, edgecolors= 'k')
plt.xlabel("X")
plt.ylabel("Y")
plt.title("Decision boundary plot of MLFFNN with a 2 hidden layer for all classes")
```

Out[ ]:

```
Text(0.5, 1.0, 'Decision boundary plot of MLFFNN with a 2 hidden layer for all classes')
```



In [ ]:

```
test_acc = best_model.score(df2_test_scaled, df2_test.label)
```

```
print("Accuracy on the test set is  ", test_acc)
```

```
Accuracy on the test set is   1.0
```

In [ ]:

```
confusion_matrix(df2.label, best_model.predict(df2_scaled))
```

Out[ ]:

```
array([[199,   0,   0],
       [  0, 200,   0],
       [  0,   0, 200]])
```

In [ ]:

```
confusion_matrix(df2_test.label, best_model.predict(df2_test_scaled))
```

Out[ ]:

```
array([[29,  0,  0],
       [ 0, 30,  0],
       [ 0,  0, 30]])
```

In [ ]:

```
test_acc = best_model.score(df2_test_scaled, df2_test.label)
print("Test accuracy for learning rate = ", 0.1, "and number of hidden units layer 1 = "
, 3 ,"and number of hidden units layer 2 = ", 3 , "is", test_acc)
```

```
Test accuracy for learning rate =  0.1 and number of hidden units layer 1 =  3 and number
of hidden units layer 2 =  3 is 1.0
```

SVM one v/s rest

RBF

In [6]:

```
df2 = pd.read_csv("/content/train (1).csv")
```

In [7]:

```
df2.columns = ["x", "y", "label"]
```

In [8]:

```
df2.head()
```

Out[8]:

	x	y	label
0	0.963906	-0.084686	0.0
1	0.751577	-0.692578	0.0
2	0.064676	-1.029066	0.0
3	0.945798	-0.525876	0.0
4	1.057655	0.000716	0.0

In [9]:

```
df_0 = df2[df2.label == 0]
df_0_others = df2[df2.label != 0]
```

In [10]:

```
df_0_others = df_0_others.assign(label=1)
```

In [11]:

```
df_0  = df_0.append(df_0_others)
```

In [12]:

```
df_0_labels = df_0.label
```

In [13]:

```
scaler = StandardScaler().fit(df_0.drop(columns=["label"]))
```

In [14]:

```
df_0_scaled = scaler.transform(df_0.drop(columns=["label"]))
```

In [15]:

```
df_1 = df2[df2.label == 1]
df_1_others = df2[df2.label != 1]
```

In [16]:

```
df_1 = df_1.assign(label = 0)
df_1_others = df_1_others.assign(label = 1)
```

In [17]:

```
df_1  = df_1.append(df_1_others)
```

In [18]:

```
df_1_labels = df_1.label
```

In [19]:

```
scaler = StandardScaler().fit(df_1.drop(columns=["label"]))
```

In [20]:

```
df_1_scaled = scaler.transform(df_1.drop(columns=["label"]))
```

In [21]:

```
df_2 = df2[df2.label == 2]
df_2_others = df2[df2.label != 2]
```

In [22]:

```
df_2 = df_2.assign(label = 0)
df_2_others = df_2_others.assign(label = 1)
```

In [23]:

```
df_2  = df_2.append(df_2_others)
```

In [24]:

```
df_2_labels = df_2.label
```

In [25]:

```
scaler = StandardScaler().fit(df_2.drop(columns=["label"]))
```

In [26]:

```
df_2_scaled = scaler.transform(df_2.drop(columns=["label"]))
```

0 v/s others

In [ ]:

```
c = [0.0001, 0.001, 0.01, 0.1, 1, 10]
```

In [ ]:

```
gamma = [0.0001, 0.001, 0.01, 0.1, 1, 10]
```

In [ ]:

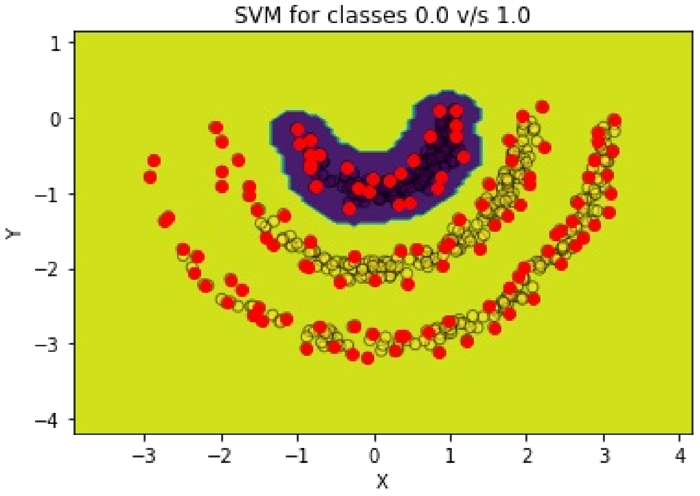
```
for i in c:
    for g in gamma:
        clf = SVC(C=i, gamma=g,  kernel='rbf')
        cv_results = cross_validate(clf, df_0_scaled, df_0_labels, cv=3)
        print("Cross Validation Test Score for C = ", i, "and gamma = ", g , "is",np.mean(cv_results['test_score']))
        train_acc = SVC(C=i, gamma=g,  kernel='rbf').fit(df_0_scaled, df_0_labels).score(df_0_scaled, df_0_labels)
        print("Train accuracy for C = ", i, "and gamma = ", g , "is", train_acc)
```

```
Cross Validation Test Score for C =  0.0001 and gamma =  0.0001 is 0.6677805695142379
Train accuracy for C =  0.0001 and gamma =  0.0001 is 0.667779632721202
Cross Validation Test Score for C =  0.0001 and gamma =  0.001 is 0.6677805695142379
Train accuracy for C =  0.0001 and gamma =  0.001 is 0.667779632721202
Cross Validation Test Score for C =  0.0001 and gamma =  0.01 is 0.6677805695142379
Train accuracy for C =  0.0001 and gamma =  0.01 is 0.667779632721202
Cross Validation Test Score for C =  0.0001 and gamma =  0.1 is 0.6677805695142379
Train accuracy for C =  0.0001 and gamma =  0.1 is 0.667779632721202
Cross Validation Test Score for C =  0.0001 and gamma =  1 is 0.6677805695142379
Train accuracy for C =  0.0001 and gamma =  1 is 0.667779632721202
Cross Validation Test Score for C =  0.0001 and gamma =  10 is 0.6677805695142379
Train accuracy for C =  0.0001 and gamma =  10 is 0.667779632721202
Cross Validation Test Score for C =  0.001 and gamma =  0.0001 is 0.6677805695142379
Train accuracy for C =  0.001 and gamma =  0.0001 is 0.667779632721202
Cross Validation Test Score for C =  0.001 and gamma =  0.001 is 0.6677805695142379
Train accuracy for C =  0.001 and gamma =  0.001 is 0.667779632721202
Cross Validation Test Score for C =  0.001 and gamma =  0.01 is 0.6677805695142379
Train accuracy for C =  0.001 and gamma =  0.01 is 0.667779632721202
Cross Validation Test Score for C =  0.001 and gamma =  0.1 is 0.6677805695142379
Train accuracy for C =  0.001 and gamma =  0.1 is 0.667779632721202
Cross Validation Test Score for C =  0.001 and gamma =  1 is 0.6677805695142379
Train accuracy for C =  0.001 and gamma =  1 is 0.667779632721202
Cross Validation Test Score for C =  0.001 and gamma =  10 is 0.6677805695142379
Train accuracy for C =  0.001 and gamma =  10 is 0.667779632721202
Cross Validation Test Score for C =  0.01 and gamma =  0.0001 is 0.6677805695142379
Train accuracy for C =  0.01 and gamma =  0.0001 is 0.667779632721202
Cross Validation Test Score for C =  0.01 and gamma =  0.001 is 0.6677805695142379
Train accuracy for C =  0.01 and gamma =  0.001 is 0.667779632721202
Cross Validation Test Score for C =  0.01 and gamma =  0.01 is 0.6677805695142379
Train accuracy for C =  0.01 and gamma =  0.01 is 0.667779632721202
Cross Validation Test Score for C =  0.01 and gamma =  0.1 is 0.6677805695142379
Train accuracy for C =  0.01 and gamma =  0.1 is 0.667779632721202
Cross Validation Test Score for C =  0.01 and gamma =  1 is 0.6827805695142378
Train accuracy for C =  0.01 and gamma =  1 is 0.9215358931552587
Cross Validation Test Score for C =  0.01 and gamma =  10 is 0.6677805695142379
Train accuracy for C =  0.01 and gamma =  10 is 0.667779632721202
Cross Validation Test Score for C =  0.1 and gamma =  0.0001 is 0.6677805695142379
Train accuracy for C =  0.1 and gamma =  0.0001 is 0.667779632721202
Cross Validation Test Score for C =  0.1 and gamma =  0.001 is 0.6677805695142379
Train accuracy for C =  0.1 and gamma =  0.001 is 0.667779632721202
Cross Validation Test Score for C =  0.1 and gamma =  0.01 is 0.6677805695142379
Train accuracy for C =  0.1 and gamma =  0.01 is 0.667779632721202
Cross Validation Test Score for C =  0.1 and gamma =  0.1 is 0.9433333333333334
Train accuracy for C =  0.1 and gamma =  0.1 is 0.9749582637729549
Cross Validation Test Score for C =  0.1 and gamma =  1 is 1.0
Train accuracy for C =  0.1 and gamma =  1 is 1.0
Cross Validation Test Score for C =  0.1 and gamma =  10 is 0.9849832495812395
```

```
Train accuracy for C =  0.1 and gamma =  10 is 0.998330550918197
Cross Validation Test Score for C =  1 and gamma =  0.0001 is 0.6677805695142379
Train accuracy for C =  1 and gamma =  0.0001 is 0.667779632721202
Cross Validation Test Score for C =  1 and gamma =  0.001 is 0.6677805695142379
Train accuracy for C =  1 and gamma =  0.001 is 0.667779632721202
Cross Validation Test Score for C =  1 and gamma =  0.01 is 0.90321608040201
Train accuracy for C =  1 and gamma =  0.01 is 0.9348914858096828
Cross Validation Test Score for C =  1 and gamma =  0.1 is 0.9916666666666667
Train accuracy for C =  1 and gamma =  0.1 is 1.0
Cross Validation Test Score for C =  1 and gamma =  1 is 1.0
Train accuracy for C =  1 and gamma =  1 is 1.0
Cross Validation Test Score for C =  1 and gamma =  10 is 1.0
Train accuracy for C =  1 and gamma =  10 is 1.0
Cross Validation Test Score for C =  10 and gamma =  0.0001 is 0.6677805695142379
Train accuracy for C =  10 and gamma =  0.0001 is 0.667779632721202
Cross Validation Test Score for C =  10 and gamma =  0.001 is 0.8915326633165829
Train accuracy for C =  10 and gamma =  0.001 is 0.9181969949916527
Cross Validation Test Score for C =  10 and gamma =  0.01 is 0.9382998324958124
Train accuracy for C =  10 and gamma =  0.01 is 0.9699499165275459
Cross Validation Test Score for C =  10 and gamma =  0.1 is 1.0
Train accuracy for C =  10 and gamma =  0.1 is 1.0
Cross Validation Test Score for C =  10 and gamma =  1 is 1.0
Train accuracy for C =  10 and gamma =  1 is 1.0
Cross Validation Test Score for C =  10 and gamma =  10 is 1.0
Train accuracy for C =  10 and gamma =  10 is 1.0
```

In [ ]:

```python
best_model = SVC(C=10, gamma=10,  kernel='rbf').fit(df_0_scaled, df_0_labels)
plot_pairs_svm(best_model, df_0)
```



1 v/s others

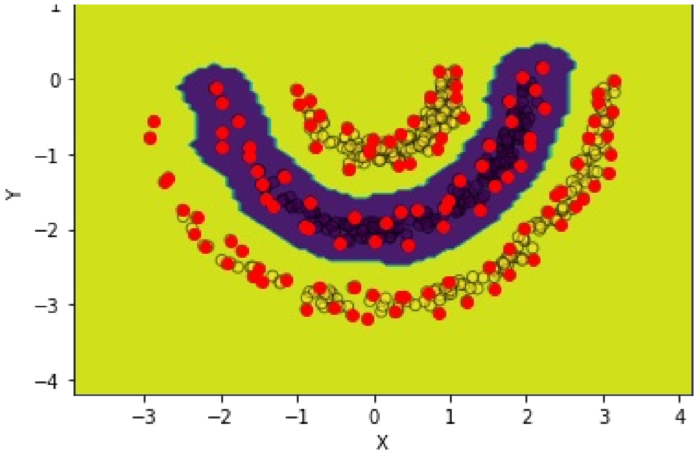
In [ ]:

```python
for i in c:
    for g in gamma:
        clf = SVC(C=i, gamma=g,  kernel='rbf')
        cv_results = cross_validate(clf, df_1_scaled, df_1_labels, cv=3)
        print("Cross Validation Test Score for C = ", i, "and gamma = ", g , "is",np.mea
n(cv_results['test_score']))
        train_acc = SVC(C=i, gamma=g,  kernel='rbf').fit(df_1_scaled, df_1_labels).score
(df_1_scaled, df_1_labels)
        print("Train accuracy for C = ", i, "and gamma = ", g , "is", train_acc)
```

```
Cross Validation Test Score for C =  0.0001 and gamma =  0.0001 is 0.6661139028475712
Train accuracy for C =  0.0001 and gamma =  0.0001 is 0.666110183639399
Cross Validation Test Score for C =  0.0001 and gamma =  0.001 is 0.6661139028475712
Train accuracy for C =  0.0001 and gamma =  0.001 is 0.666110183639399
Cross Validation Test Score for C =  0.0001 and gamma =  0.01 is 0.6661139028475712
Train accuracy for C =  0.0001 and gamma =  0.01 is 0.666110183639399
Cross Validation Test Score for C =  0.0001 and gamma =  0.1 is 0.6661139028475712
Train accuracy for C =  0.0001 and gamma =  0.1 is 0.666110183639399
Train accuracy for C =  0.0001 and gamma =  0.1 is 0.666110183639399
Cross Validation Test Score for C =  0.0001 and gamma =  1 is 0.6661139028475712
Train accuracy for C =  0.0001 and gamma =  1 is 0.666110183639399
Cross Validation Test Score for C =  0.0001 and gamma =  10 is 0.6661139028475712
Train accuracy for C =  0.0001 and gamma =  10 is 0.666110183639399
Cross Validation Test Score for C =  0.001 and gamma =  0.0001 is 0.6661139028475712
Train accuracy for C =  0.001 and gamma =  0.0001 is 0.666110183639399
Cross Validation Test Score for C =  0.001 and gamma =  0.001 is 0.6661139028475712
Train accuracy for C =  0.001 and gamma =  0.001 is 0.666110183639399
Cross Validation Test Score for C =  0.001 and gamma =  0.01 is 0.6661139028475712
Train accuracy for C =  0.001 and gamma =  0.01 is 0.666110183639399
Cross Validation Test Score for C =  0.001 and gamma =  0.1 is 0.6661139028475712
Train accuracy for C =  0.001 and gamma =  0.1 is 0.666110183639399
Cross Validation Test Score for C =  0.001 and gamma =  1 is 0.6661139028475712
Train accuracy for C =  0.001 and gamma =  1 is 0.666110183639399
Cross Validation Test Score for C =  0.001 and gamma =  10 is 0.6661139028475712
Train accuracy for C =  0.001 and gamma =  10 is 0.666110183639399
Cross Validation Test Score for C =  0.01 and gamma =  0.0001 is 0.6661139028475712
Train accuracy for C =  0.01 and gamma =  0.0001 is 0.666110183639399
Cross Validation Test Score for C =  0.01 and gamma =  0.001 is 0.6661139028475712
Train accuracy for C =  0.01 and gamma =  0.001 is 0.666110183639399
Cross Validation Test Score for C =  0.01 and gamma =  0.01 is 0.6661139028475712
Train accuracy for C =  0.01 and gamma =  0.01 is 0.666110183639399
Cross Validation Test Score for C =  0.01 and gamma =  0.1 is 0.6661139028475712
Train accuracy for C =  0.01 and gamma =  0.1 is 0.666110183639399
Cross Validation Test Score for C =  0.01 and gamma =  1 is 0.6661139028475712
Train accuracy for C =  0.01 and gamma =  1 is 0.666110183639399
Cross Validation Test Score for C =  0.01 and gamma =  10 is 0.6661139028475712
Train accuracy for C =  0.01 and gamma =  10 is 0.666110183639399
Cross Validation Test Score for C =  0.1 and gamma =  0.0001 is 0.6661139028475712
Train accuracy for C =  0.1 and gamma =  0.0001 is 0.666110183639399
Cross Validation Test Score for C =  0.1 and gamma =  0.001 is 0.6661139028475712
Train accuracy for C =  0.1 and gamma =  0.001 is 0.666110183639399
Cross Validation Test Score for C =  0.1 and gamma =  0.01 is 0.6661139028475712
Train accuracy for C =  0.1 and gamma =  0.01 is 0.666110183639399
Cross Validation Test Score for C =  0.1 and gamma =  0.1 is 0.6661139028475712
Train accuracy for C =  0.1 and gamma =  0.1 is 0.666110183639399
Cross Validation Test Score for C =  0.1 and gamma =  1 is 0.9816499162479063
Train accuracy for C =  0.1 and gamma =  1 is 0.993322203672788
Cross Validation Test Score for C =  0.1 and gamma =  10 is 0.9766499162479061
Train accuracy for C =  0.1 and gamma =  10 is 0.986644407345576
Cross Validation Test Score for C =  1 and gamma =  0.0001 is 0.6661139028475712
Train accuracy for C =  1 and gamma =  0.0001 is 0.666110183639399
Cross Validation Test Score for C =  1 and gamma =  0.001 is 0.6661139028475712
Train accuracy for C =  1 and gamma =  0.001 is 0.666110183639399
Cross Validation Test Score for C =  1 and gamma =  0.01 is 0.6661139028475712
Train accuracy for C =  1 and gamma =  0.01 is 0.666110183639399
Cross Validation Test Score for C =  1 and gamma =  0.1 is 0.7812981574539363
Train accuracy for C =  1 and gamma =  0.1 is 0.8981636060100167
Cross Validation Test Score for C =  1 and gamma =  1 is 0.9966499162479062
Train accuracy for C =  1 and gamma =  1 is 1.0
Cross Validation Test Score for C =  1 and gamma =  10 is 1.0
Train accuracy for C =  1 and gamma =  10 is 1.0
Cross Validation Test Score for C =  10 and gamma =  0.0001 is 0.6661139028475712
Train accuracy for C =  10 and gamma =  0.0001 is 0.666110183639399
Cross Validation Test Score for C =  10 and gamma =  0.001 is 0.6661139028475712
Train accuracy for C =  10 and gamma =  0.001 is 0.666110183639399
Cross Validation Test Score for C =  10 and gamma =  0.01 is 0.6661139028475712
Train accuracy for C =  10 and gamma =  0.01 is 0.666110183639399
Cross Validation Test Score for C =  10 and gamma =  0.1 is 0.9348576214405361
Train accuracy for C =  10 and gamma =  0.1 is 0.9833055091819699
Cross Validation Test Score for C =  10 and gamma =  1 is 0.9966499162479062
Train accuracy for C =  10 and gamma =  1 is 1.0
Cross Validation Test Score for C =  10 and gamma =  10 is 1.0
Train accuracy for C =  10 and gamma =  10 is 1.0
```

In [ ]:

```
best_model = SVC(C=10, gamma=10,  kernel='rbf').fit(df_1_scaled, df_1_labels)
plot_pairs_svm(best_model, df_1)
```



2 v/s others

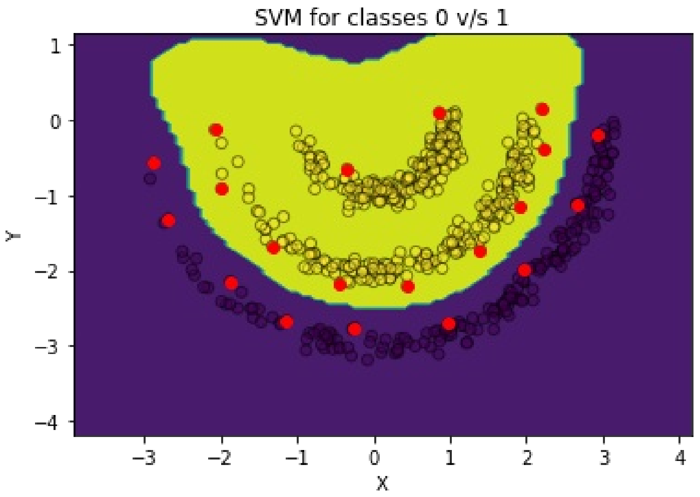
In [ ]:

```
for i in c:
    for g in gamma:
        clf = SVC(C=i, gamma=g,  kernel='rbf')
        cv_results = cross_validate(clf, df_2_scaled, df_2_labels, cv=3)
        print("Cross Validation Test Score for C = ", i, "and gamma = ", g , "is",np.mean(cv_results['test_score']))
        train_acc = SVC(C=i, gamma=g,  kernel='rbf').fit(df_2_scaled, df_2_labels).score(df_2_scaled, df_2_labels)
        print("Train accuracy for C = ", i, "and gamma = ", g , "is", train_acc)
```

```
Cross Validation Test Score for C =  0.0001 and gamma =  0.0001 is 0.6661139028475712
Train accuracy for C =  0.0001 and gamma =  0.0001 is 0.666110183639399
Cross Validation Test Score for C =  0.0001 and gamma =  0.001 is 0.6661139028475712
Train accuracy for C =  0.0001 and gamma =  0.001 is 0.666110183639399
Cross Validation Test Score for C =  0.0001 and gamma =  0.01 is 0.6661139028475712
Train accuracy for C =  0.0001 and gamma =  0.01 is 0.666110183639399
Cross Validation Test Score for C =  0.0001 and gamma =  0.1 is 0.6661139028475712
Train accuracy for C =  0.0001 and gamma =  0.1 is 0.666110183639399
Cross Validation Test Score for C =  0.0001 and gamma =  1 is 0.6661139028475712
Train accuracy for C =  0.0001 and gamma =  1 is 0.666110183639399
Cross Validation Test Score for C =  0.0001 and gamma =  10 is 0.6661139028475712
Train accuracy for C =  0.0001 and gamma =  10 is 0.666110183639399
Cross Validation Test Score for C =  0.001 and gamma =  0.0001 is 0.6661139028475712
Train accuracy for C =  0.001 and gamma =  0.0001 is 0.666110183639399
Cross Validation Test Score for C =  0.001 and gamma =  0.001 is 0.6661139028475712
Train accuracy for C =  0.001 and gamma =  0.001 is 0.666110183639399
Cross Validation Test Score for C =  0.001 and gamma =  0.01 is 0.6661139028475712
Train accuracy for C =  0.001 and gamma =  0.01 is 0.666110183639399
Cross Validation Test Score for C =  0.001 and gamma =  0.1 is 0.6661139028475712
Train accuracy for C =  0.001 and gamma =  0.1 is 0.666110183639399
Cross Validation Test Score for C =  0.001 and gamma =  1 is 0.6661139028475712
Train accuracy for C =  0.001 and gamma =  1 is 0.666110183639399
Cross Validation Test Score for C =  0.001 and gamma =  10 is 0.6661139028475712
Train accuracy for C =  0.001 and gamma =  10 is 0.666110183639399
Cross Validation Test Score for C =  0.01 and gamma =  0.0001 is 0.6661139028475712
Train accuracy for C =  0.01 and gamma =  0.0001 is 0.666110183639399
Cross Validation Test Score for C =  0.01 and gamma =  0.001 is 0.6661139028475712
Train accuracy for C =  0.01 and gamma =  0.001 is 0.666110183639399
Cross Validation Test Score for C =  0.01 and gamma =  0.01 is 0.6661139028475712
Train accuracy for C =  0.01 and gamma =  0.01 is 0.666110183639399
Cross Validation Test Score for C =  0.01 and gamma =  0.1 is 0.6661139028475712
Train accuracy for C =  0.01 and gamma =  0.1 is 0.666110183639399
Cross Validation Test Score for C =  0.01 and gamma =  1 is 0.6661139028475712
Train accuracy for C =  0.01 and gamma =  1 is 0.666110183639399
Cross Validation Test Score for C =  0.01 and gamma =  10 is 0.6661139028475712
Train accuracy for C =  0.01 and gamma =  10 is 0.666110183639399
Cross Validation Test Score for C =  0.1 and gamma =  0.0001 is 0.6661139028475712
Train accuracy for C =  0.1 and gamma =  0.0001 is 0.666110183639399
Cross Validation Test Score for C =  0.1 and gamma =  0.001 is 0.6661139028475712
Train accuracy for C =  0.1 and gamma =  0.001 is 0.666110183639399
Cross Validation Test Score for C =  0.1 and gamma =  0.01 is 0.6661139028475712
Cross Validation Test Score for C =  0.1 and gamma =  0.01 is 0.6661139028475712
Train accuracy for C =  0.1 and gamma =  0.01 is 0.666110183639399
Cross Validation Test Score for C =  0.1 and gamma =  0.1 is 0.951465661641541
Train accuracy for C =  0.1 and gamma =  0.1 is 0.994991652754591
Cross Validation Test Score for C =  0.1 and gamma =  1 is 0.9966499162479062
Train accuracy for C =  0.1 and gamma =  1 is 0.998330550918197
Cross Validation Test Score for C =  0.1 and gamma =  10 is 0.9899664991624791
Train accuracy for C =  0.1 and gamma =  10 is 0.996661101836394
Cross Validation Test Score for C =  1 and gamma =  0.0001 is 0.6661139028475712
Train accuracy for C =  1 and gamma =  0.0001 is 0.666110183639399
Cross Validation Test Score for C =  1 and gamma =  0.001 is 0.6661139028475712
Train accuracy for C =  1 and gamma =  0.001 is 0.666110183639399
Cross Validation Test Score for C =  1 and gamma =  0.01 is 0.8495728643216082
Train accuracy for C =  1 and gamma =  0.01 is 0.9131886477462438
Cross Validation Test Score for C =  1 and gamma =  0.1 is 1.0
Train accuracy for C =  1 and gamma =  0.1 is 1.0
Cross Validation Test Score for C =  1 and gamma =  1 is 0.9966499162479062
Train accuracy for C =  1 and gamma =  1 is 1.0
Cross Validation Test Score for C =  1 and gamma =  10 is 0.9932998324958123
Train accuracy for C =  1 and gamma =  10 is 1.0
Cross Validation Test Score for C =  10 and gamma =  0.0001 is 0.6661139028475712
Train accuracy for C =  10 and gamma =  0.0001 is 0.666110183639399
Cross Validation Test Score for C =  10 and gamma =  0.001 is 0.8428894472361809
Train accuracy for C =  10 and gamma =  0.001 is 0.9031719532554258
Cross Validation Test Score for C =  10 and gamma =  0.01 is 0.9866247906197655
Train accuracy for C =  10 and gamma =  0.01 is 0.993322203672788
Cross Validation Test Score for C =  10 and gamma =  0.1 is 1.0
Train accuracy for C =  10 and gamma =  0.1 is 1.0
Cross Validation Test Score for C =  10 and gamma =  1 is 0.9966499162479062
Train accuracy for C =  10 and gamma =  1 is 1.0
Cross Validation Test Score for C =  10 and gamma =  10 is 0.9932998324958123
Train accuracy for C =  10 and gamma =  10 is 1.0
```

In [ ]:

```
best_model = SVC(C=10, gamma=1,  kernel='rbf').fit(df_2_scaled, df_2_labels)
plot_pairs_svm(best_model, df_2)
```



Combining all models using oneVsRestClassifier

In [ ]:

```
c = [0.0001, 0.001, 0.01, 0.1, 1, 10]
```

In [ ]:

```
gamma = [0.0001, 0.001, 0.01, 0.1, 1, 10]
```

In [ ]:

```
for i in c:
    for g in gamma:
        clf = OneVsRestClassifier(SVC(C=i, gamma=g,  kernel='rbf'))
        cv_results = cross_validate(clf, df2_scaled, df2.label, cv=3)
        print("Cross Validation Test Score for C = ", i, "and gamma = ", g , "is",np.mea
n(cv_results['test_score']))
        train_acc = OneVsRestClassifier(SVC(C=i, gamma=g,  kernel='rbf')).fit(df2_scaled
, df2.label).score(df2_scaled, df2.label)
        print("Train accuracy for C = ", i, "and gamma = ", g , "is",train_acc)
```

```
Cross Validation Test Score for C =  0.0001 and gamma =  0.0001 is 0.6343802345058626
Train accuracy for C =  0.0001 and gamma =  0.0001 is 0.6310517529215359
Cross Validation Test Score for C =  0.0001 and gamma =  0.001 is 0.6377219430485762
Train accuracy for C =  0.0001 and gamma =  0.001 is 0.6310517529215359
Cross Validation Test Score for C =  0.0001 and gamma =  0.01 is 0.6493969849246232
Train accuracy for C =  0.0001 and gamma =  0.01 is 0.6360601001669449
Cross Validation Test Score for C =  0.0001 and gamma =  0.1 is 0.6610804020100502
Train accuracy for C =  0.0001 and gamma =  0.1 is 0.662771285475793
Cross Validation Test Score for C =  0.0001 and gamma =  1 is 0.9666415410385261
Train accuracy for C =  0.0001 and gamma =  1 is 0.9732888146911519
Cross Validation Test Score for C =  0.0001 and gamma =  10 is 0.993324958123953
Train accuracy for C =  0.0001 and gamma =  10 is 1.0
Cross Validation Test Score for C =  0.001 and gamma =  0.0001 is 0.6343802345058626
Train accuracy for C =  0.001 and gamma =  0.0001 is 0.6310517529215359
Cross Validation Test Score for C =  0.001 and gamma =  0.001 is 0.6377219430485762
Train accuracy for C =  0.001 and gamma =  0.001 is 0.6310517529215359
Cross Validation Test Score for C =  0.001 and gamma =  0.01 is 0.6410636515912899
Train accuracy for C =  0.001 and gamma =  0.01 is 0.6360601001669449
Cross Validation Test Score for C =  0.001 and gamma =  0.1 is 0.6610804020100502
Train accuracy for C =  0.001 and gamma =  0.1 is 0.662771285475793
Cross Validation Test Score for C =  0.001 and gamma =  1 is 0.9683082077051927
Train accuracy for C =  0.001 and gamma =  1 is 0.9749582637729549
Cross Validation Test Score for C =  0.001 and gamma =  10 is 0.993324958123953
Train accuracy for C =  0.001 and gamma =  10 is 1.0
Cross Validation Test Score for C =  0.01 and gamma =  0.0001 is 0.6343802345058626
Train accuracy for C =  0.01 and gamma =  0.0001 is 0.6310517529215359
Cross Validation Test Score for C =  0.01 and gamma =  0.001 is 0.6293634840871022
Train accuracy for C =  0.01 and gamma =  0.001 is 0.6310517529215359
Cross Validation Test Score for C =  0.01 and gamma =  0.01 is 0.6360552763819095
Train accuracy for C =  0.01 and gamma =  0.01 is 0.6377295492487479
Cross Validation Test Score for C =  0.01 and gamma =  0.1 is 0.6610804020100502
Train accuracy for C =  0.01 and gamma =  0.1 is 0.662771285475793
Cross Validation Test Score for C =  0.01 and gamma =  1 is 0.9683082077051927
Train accuracy for C =  0.01 and gamma =  1 is 0.9749582637729549
Cross Validation Test Score for C =  0.01 and gamma =  10 is 0.993324958123953
Train accuracy for C =  0.01 and gamma =  10 is 1.0
Cross Validation Test Score for C =  0.1 and gamma =  0.0001 is 0.6293634840871022
Train accuracy for C =  0.1 and gamma =  0.0001 is 0.6310517529215359
Cross Validation Test Score for C =  0.1 and gamma =  0.001 is 0.6260301507537688
Train accuracy for C =  0.1 and gamma =  0.001 is 0.6260434056761269
Cross Validation Test Score for C =  0.1 and gamma =  0.01 is 0.6377303182579565
Train accuracy for C =  0.1 and gamma =  0.01 is 0.6360601001669449
Cross Validation Test Score for C =  0.1 and gamma =  0.1 is 0.6710887772194304
Train accuracy for C =  0.1 and gamma =  0.1 is 0.7696160267111853
Cross Validation Test Score for C =  0.1 and gamma =  1 is 0.995
Train accuracy for C =  0.1 and gamma =  1 is 1.0
Cross Validation Test Score for C =  0.1 and gamma =  10 is 0.9899916247906199
Train accuracy for C =  0.1 and gamma =  10 is 1.0
Cross Validation Test Score for C =  1 and gamma =  0.0001 is 0.6243634840871022
Train accuracy for C =  1 and gamma =  0.0001 is 0.6243739565943238
Cross Validation Test Score for C =  1 and gamma =  0.001 is 0.6277051926298157
Train accuracy for C =  1 and gamma =  0.001 is 0.6260434056761269
Cross Validation Test Score for C =  1 and gamma =  0.01 is 0.6360469011725294
Train accuracy for C =  1 and gamma =  0.01 is 0.6460767946577629
Cross Validation Test Score for C =  1 and gamma =  0.1 is 0.9599581239530988
Train accuracy for C =  1 and gamma =  0.1 is 0.9732888146911519
Cross Validation Test Score for C =  1 and gamma =  1 is 0.9966499162479062
Train accuracy for C =  1 and gamma =  1 is 1.0
Cross Validation Test Score for C =  1 and gamma =  10 is 1.0
Train accuracy for C =  1 and gamma =  10 is 1.0
Cross Validation Test Score for C =  10 and gamma =  0.0001 is 0.626038525963149
Train accuracy for C =  10 and gamma =  0.0001 is 0.6243739565943238
Cross Validation Test Score for C =  10 and gamma =  0.001 is 0.6193467336683417
Train accuracy for C =  10 and gamma =  0.001 is 0.6193656093489148
```

```
Cross Validation Test Score for C =  10 and gamma =  0.01 is 0.7779480737018426
Train accuracy for C =  10 and gamma =  0.01 is 0.8464106844741235
Cross Validation Test Score for C =  10 and gamma =  0.1 is 0.9799497487437185
Train accuracy for C =  10 and gamma =  0.1 is 0.991652754590985
Cross Validation Test Score for C =  10 and gamma =  1 is 0.9966499162479062
Train accuracy for C =  10 and gamma =  1 is 1.0
Cross Validation Test Score for C =  10 and gamma =  10 is 1.0
Train accuracy for C =  10 and gamma =  10 is 1.0
```

In [ ]:

```
best_model  = OneVsRestClassifier(SVC(C=10, gamma=10,  kernel='rbf')).fit(df2_scaled, df
2.label)
```

In [ ]:

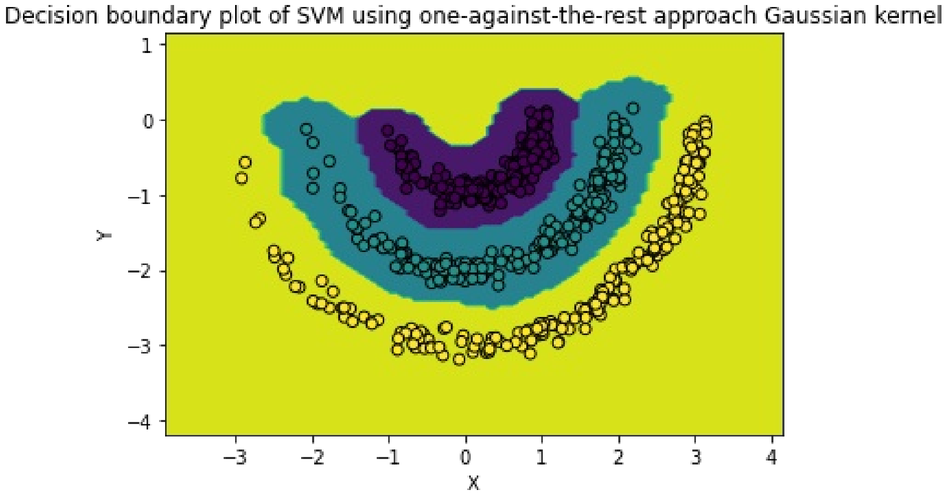
```
X = df2[['x','y']].values
```

In [ ]:

```
xmin, xmax = np.min(X[:,0]) - 1, np.max(X[:,0]) + 1
ymin, ymax = np.min(X[:,1]) - 1, np.max(X[:,1]) + 1

x = np.linspace(xmin,xmax,100)
y = np.linspace(ymin,ymax,100)

xx, yy = np.meshgrid(x,y)

Z = np.stack((xx.ravel(),yy.ravel()),axis = 1)

Z = scaler.transform(Z)
pred = best_model.predict(Z).reshape(100,100)
# print(pred.shape)

plt.contourf(xx,yy,pred)
plt.scatter(X[:,0],X[:,1],c = df2.label, edgecolors= 'k')
plt.xlabel("X")
plt.ylabel("Y")
plt.title("Decision boundary plot of SVM using one-against-the-rest approach Gaussian ker
nel")
```

Out[ ]:

```
Text(0.5, 1.0, 'Decision boundary plot of SVM using one-against-the-rest approach Gaussia
n kernel')
```



In [ ]:

```
confusion_matrix(df2.label, best_model.predict(df2_scaled))
```

Out[ ]:

```
array([[199,   0,   0],
       [  0, 200,   0],
       [  0,   0, 200]])
```

In [ ]:

```
confusion_matrix(df2_test.label, best_model.predict(df2_test_scaled))
```

Out[ ]:

```
array([[29,  0,  0],
       [ 0, 30,  0],
       [ 0,  0, 30]])
```

In [ ]:

```
test_acc = best_model.score(df2_test_scaled, df2_test.label)
print("Test accuracy for C = ", 10, "and gamma = ", 10 , "is",test_acc)
```

```
Test accuracy for C =  10 and gamma =  10 is 1.0
```

POLYNOMIAL

0 v/s others

In [32]:

```
c = [0.0001, 0.001, 0.01, 0.1, 1, 10]
```

In [33]:

```
gamma = [0.0001, 0.001, 0.01, 0.1, 1]
```

In [36]:

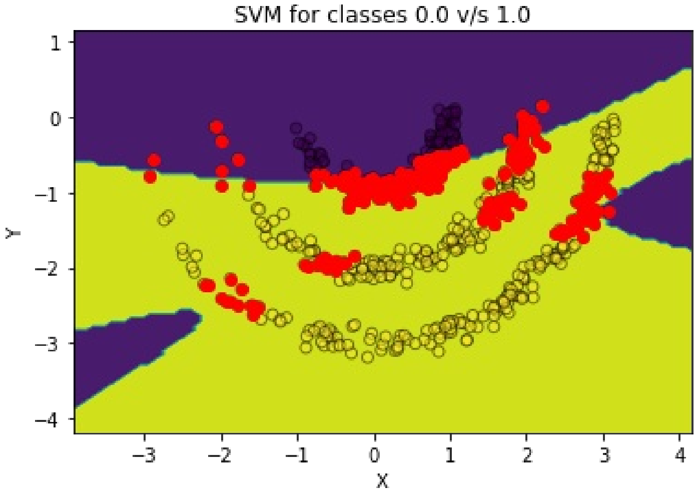
```
for i in c:
    for g in gamma:
        clf = SVC(C=i, gamma=g,  kernel='poly')
        cv_results = cross_validate(clf, df_0_scaled, df_0_labels, cv=3)
        print("Cross Validation Test Score for C = ", i, "and gamma = ", g , "is",np.mea
n(cv_results['test_score']))
        train_acc = SVC(C=i, gamma=g,  kernel='poly').fit(df_0_scaled, df_0_labels).scor
e(df_0_scaled, df_0_labels)
        print("Train accuracy for C = ", i, "and gamma = ", g , "is", train_acc)
```

```
Cross Validation Test Score for C =  0.0001 and gamma =  0.0001 is 0.6677805695142379
Train accuracy for C =  0.0001 and gamma =  0.0001 is 0.667779632721202
Cross Validation Test Score for C =  0.0001 and gamma =  0.001 is 0.6677805695142379
Train accuracy for C =  0.0001 and gamma =  0.001 is 0.667779632721202
Cross Validation Test Score for C =  0.0001 and gamma =  0.01 is 0.6677805695142379
Train accuracy for C =  0.0001 and gamma =  0.01 is 0.667779632721202
Cross Validation Test Score for C =  0.0001 and gamma =  0.1 is 0.6677805695142379
Train accuracy for C =  0.0001 and gamma =  0.1 is 0.667779632721202
Cross Validation Test Score for C =  0.0001 and gamma =  1 is 0.6677805695142379
Train accuracy for C =  0.0001 and gamma =  1 is 0.667779632721202
Cross Validation Test Score for C =  0.001 and gamma =  0.0001 is 0.6677805695142379
Train accuracy for C =  0.001 and gamma =  0.0001 is 0.667779632721202
Cross Validation Test Score for C =  0.001 and gamma =  0.001 is 0.6677805695142379
Train accuracy for C =  0.001 and gamma =  0.001 is 0.667779632721202
Cross Validation Test Score for C =  0.001 and gamma =  0.01 is 0.6677805695142379
Train accuracy for C =  0.001 and gamma =  0.01 is 0.667779632721202
Cross Validation Test Score for C =  0.001 and gamma =  0.1 is 0.6677805695142379
Train accuracy for C =  0.001 and gamma =  0.1 is 0.667779632721202
Cross Validation Test Score for C =  0.001 and gamma =  1 is 0.6677805695142379
Train accuracy for C =  0.001 and gamma =  1 is 0.667779632721202
Cross Validation Test Score for C =  0.01 and gamma =  0.0001 is 0.6677805695142379
Train accuracy for C =  0.01 and gamma =  0.0001 is 0.667779632721202
Cross Validation Test Score for C =  0.01 and gamma =  0.001 is 0.6677805695142379
Train accuracy for C =  0.01 and gamma =  0.001 is 0.667779632721202
Cross Validation Test Score for C =  0.01 and gamma =  0.01 is 0.6677805695142379
Train accuracy for C =  0.01 and gamma =  0.01 is 0.667779632721202
Cross Validation Test Score for C =  0.01 and gamma =  0.1 is 0.6677805695142379
```

```
Train accuracy for C =  0.01 and gamma =  0.1 is 0.667779632721202
Cross Validation Test Score for C =  0.01 and gamma =  1 is 0.73285594639866
Train accuracy for C =  0.01 and gamma =  1 is 0.7495826377295493
Cross Validation Test Score for C =  0.1 and gamma =  0.0001 is 0.6677805695142379
Train accuracy for C =  0.1 and gamma =  0.0001 is 0.667779632721202
Cross Validation Test Score for C =  0.1 and gamma =  0.001 is 0.6677805695142379
Train accuracy for C =  0.1 and gamma =  0.001 is 0.667779632721202
Cross Validation Test Score for C =  0.1 and gamma =  0.01 is 0.6677805695142379
Train accuracy for C =  0.1 and gamma =  0.01 is 0.667779632721202
Cross Validation Test Score for C =  0.1 and gamma =  0.1 is 0.6677805695142379
Train accuracy for C =  0.1 and gamma =  0.1 is 0.667779632721202
Cross Validation Test Score for C =  0.1 and gamma =  1 is 0.7879564489112229
Train accuracy for C =  0.1 and gamma =  1 is 0.8030050083472454
Cross Validation Test Score for C =  1 and gamma =  0.0001 is 0.6677805695142379
Train accuracy for C =  1 and gamma =  0.0001 is 0.667779632721202
Cross Validation Test Score for C =  1 and gamma =  0.001 is 0.6677805695142379
Train accuracy for C =  1 and gamma =  0.001 is 0.667779632721202
Cross Validation Test Score for C =  1 and gamma =  0.01 is 0.6677805695142379
Train accuracy for C =  1 and gamma =  0.01 is 0.667779632721202
Cross Validation Test Score for C =  1 and gamma =  0.1 is 0.6677805695142379
Train accuracy for C =  1 and gamma =  0.1 is 0.667779632721202
Cross Validation Test Score for C =  1 and gamma =  1 is 0.7979648241206031
Train accuracy for C =  1 and gamma =  1 is 0.8213689482470785
Cross Validation Test Score for C =  10 and gamma =  0.0001 is 0.6677805695142379
Train accuracy for C =  10 and gamma =  0.0001 is 0.667779632721202
Cross Validation Test Score for C =  10 and gamma =  0.001 is 0.6677805695142379
Train accuracy for C =  10 and gamma =  0.001 is 0.667779632721202
Cross Validation Test Score for C =  10 and gamma =  0.01 is 0.6677805695142379
Train accuracy for C =  10 and gamma =  0.01 is 0.667779632721202
Cross Validation Test Score for C =  10 and gamma =  0.1 is 0.73285594639866
Train accuracy for C =  10 and gamma =  0.1 is 0.7495826377295493
Cross Validation Test Score for C =  10 and gamma =  1 is 0.7879396984924624
Train accuracy for C =  10 and gamma =  1 is 0.8230383973288815
```

In [35]:

```
best_model = SVC(C=10, gamma=1,  kernel='poly').fit(df_0_scaled, df_0_labels)
plot_pairs_svm(best_model, df_0)
```



1 v/s others

In [37]:

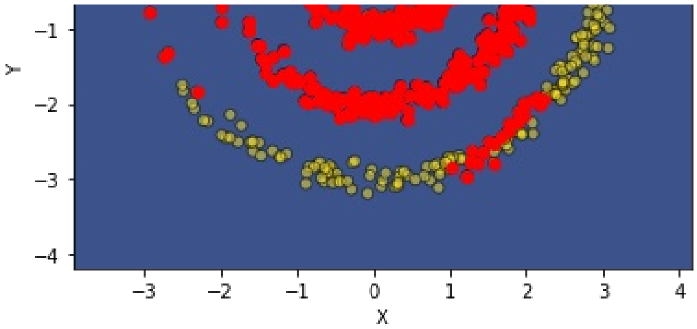
```
for i in c:
    for g in gamma:
        clf = SVC(C=i, gamma=g,  kernel='poly')
        cv_results = cross_validate(clf, df_1_scaled, df_1_labels, cv=3)
        print("Cross Validation Test Score for C = ", i, "and gamma = ", g , "is",np.mea
n(cv_results['test_score']))
        train_acc = SVC(C=i, gamma=g,  kernel='poly').fit(df_1_scaled, df_1_labels).scor
e(df_1_scaled, df_1_labels)
        print("Train accuracy for C = ", i, "and gamma = ", g , "is", train_acc)
```

```
Cross Validation Test Score for C =  0.0001 and gamma =  0.0001 is 0.6661139028475712
Train accuracy for C =  0.0001 and gamma =  0.0001 is 0.666110183639399
Cross Validation Test Score for C =  0.0001 and gamma =  0.001 is 0.6661139028475712
Train accuracy for C =  0.0001 and gamma =  0.001 is 0.666110183639399
Cross Validation Test Score for C =  0.0001 and gamma =  0.01 is 0.6661139028475712
Train accuracy for C =  0.0001 and gamma =  0.01 is 0.666110183639399
Cross Validation Test Score for C =  0.0001 and gamma =  0.1 is 0.6661139028475712
Train accuracy for C =  0.0001 and gamma =  0.1 is 0.666110183639399
Cross Validation Test Score for C =  0.0001 and gamma =  1 is 0.6661139028475712
Train accuracy for C =  0.0001 and gamma =  1 is 0.666110183639399
Cross Validation Test Score for C =  0.001 and gamma =  0.0001 is 0.6661139028475712
Train accuracy for C =  0.001 and gamma =  0.0001 is 0.666110183639399
Cross Validation Test Score for C =  0.001 and gamma =  0.001 is 0.6661139028475712
Train accuracy for C =  0.001 and gamma =  0.001 is 0.666110183639399
Cross Validation Test Score for C =  0.001 and gamma =  0.01 is 0.6661139028475712
Train accuracy for C =  0.001 and gamma =  0.01 is 0.666110183639399
Cross Validation Test Score for C =  0.001 and gamma =  0.1 is 0.6661139028475712
Train accuracy for C =  0.001 and gamma =  0.1 is 0.666110183639399
Cross Validation Test Score for C =  0.001 and gamma =  1 is 0.6661139028475712
Train accuracy for C =  0.001 and gamma =  1 is 0.666110183639399
Cross Validation Test Score for C =  0.01 and gamma =  0.0001 is 0.6661139028475712
Train accuracy for C =  0.01 and gamma =  0.0001 is 0.666110183639399
Cross Validation Test Score for C =  0.01 and gamma =  0.001 is 0.6661139028475712
Train accuracy for C =  0.01 and gamma =  0.001 is 0.666110183639399
Cross Validation Test Score for C =  0.01 and gamma =  0.01 is 0.6661139028475712
Train accuracy for C =  0.01 and gamma =  0.01 is 0.666110183639399
Cross Validation Test Score for C =  0.01 and gamma =  0.1 is 0.6661139028475712
Train accuracy for C =  0.01 and gamma =  0.1 is 0.666110183639399
Cross Validation Test Score for C =  0.01 and gamma =  1 is 0.652713567839196
Train accuracy for C =  0.01 and gamma =  1 is 0.666110183639399
Cross Validation Test Score for C =  0.1 and gamma =  0.0001 is 0.6661139028475712
Train accuracy for C =  0.1 and gamma =  0.0001 is 0.666110183639399
Cross Validation Test Score for C =  0.1 and gamma =  0.001 is 0.6661139028475712
Train accuracy for C =  0.1 and gamma =  0.001 is 0.666110183639399
Cross Validation Test Score for C =  0.1 and gamma =  0.01 is 0.6661139028475712
Train accuracy for C =  0.1 and gamma =  0.01 is 0.666110183639399
Cross Validation Test Score for C =  0.1 and gamma =  0.1 is 0.6661139028475712
Train accuracy for C =  0.1 and gamma =  0.1 is 0.666110183639399
Cross Validation Test Score for C =  0.1 and gamma =  1 is 0.652713567839196
Train accuracy for C =  0.1 and gamma =  1 is 0.666110183639399
Cross Validation Test Score for C =  1 and gamma =  0.0001 is 0.6661139028475712
Train accuracy for C =  1 and gamma =  0.0001 is 0.666110183639399
Cross Validation Test Score for C =  1 and gamma =  0.001 is 0.6661139028475712
Train accuracy for C =  1 and gamma =  0.001 is 0.666110183639399
Cross Validation Test Score for C =  1 and gamma =  0.01 is 0.6661139028475712
Train accuracy for C =  1 and gamma =  0.01 is 0.666110183639399
Cross Validation Test Score for C =  1 and gamma =  0.1 is 0.6661139028475712
Train accuracy for C =  1 and gamma =  0.1 is 0.666110183639399
Cross Validation Test Score for C =  1 and gamma =  1 is 0.652713567839196
Train accuracy for C =  1 and gamma =  1 is 0.666110183639399
Cross Validation Test Score for C =  10 and gamma =  0.0001 is 0.6661139028475712
Train accuracy for C =  10 and gamma =  0.0001 is 0.666110183639399
Cross Validation Test Score for C =  10 and gamma =  0.001 is 0.6661139028475712
Train accuracy for C =  10 and gamma =  0.001 is 0.666110183639399
Cross Validation Test Score for C =  10 and gamma =  0.01 is 0.6661139028475712
Train accuracy for C =  10 and gamma =  0.01 is 0.666110183639399
Cross Validation Test Score for C =  10 and gamma =  0.1 is 0.652713567839196
Train accuracy for C =  10 and gamma =  0.1 is 0.666110183639399
Cross Validation Test Score for C =  10 and gamma =  1 is 0.652713567839196
Train accuracy for C =  10 and gamma =  1 is 0.666110183639399
```

In [42]:

```
best_model = SVC(C=10, gamma=0.01,  kernel='poly').fit(df_1_scaled, df_1_labels)
plot_pairs_svm(best_model, df_1)
```

SVM for classes 0 v/s 1
1
0



2 v/s others

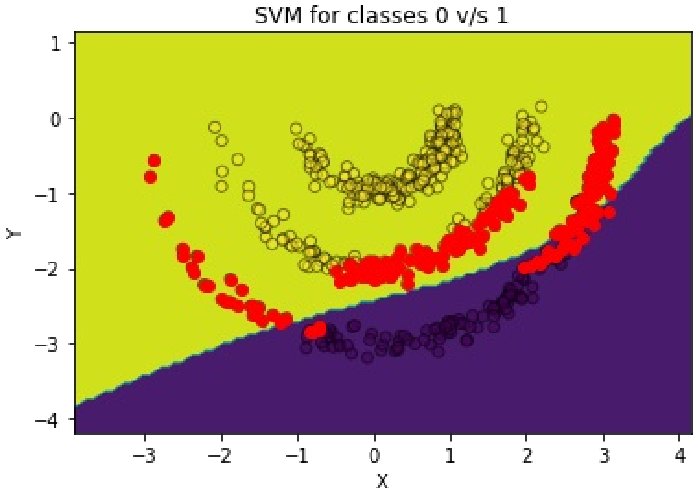
In [40]:

```
for i in c:
    for g in gamma:
        clf = SVC(C=i, gamma=g,  kernel='poly')
        cv_results = cross_validate(clf, df_2_scaled, df_2_labels, cv=3)
        print("Cross Validation Test Score for C = ", i, "and gamma = ", g , "is",np.mea
n(cv_results['test_score']))
        train_acc = SVC(C=i, gamma=g,  kernel='poly').fit(df_2_scaled, df_2_labels).scor
e(df_2_scaled, df_2_labels)
        print("Train accuracy for C = ", i, "and gamma = ", g , "is", train_acc)
```

```
Cross Validation Test Score for C =  0.0001 and gamma =  0.0001 is 0.6661139028475712
Train accuracy for C =  0.0001 and gamma =  0.0001 is 0.666110183639399
Cross Validation Test Score for C =  0.0001 and gamma =  0.001 is 0.6661139028475712
Train accuracy for C =  0.0001 and gamma =  0.001 is 0.666110183639399
Cross Validation Test Score for C =  0.0001 and gamma =  0.01 is 0.6661139028475712
Train accuracy for C =  0.0001 and gamma =  0.01 is 0.666110183639399
Cross Validation Test Score for C =  0.0001 and gamma =  0.1 is 0.6661139028475712
Train accuracy for C =  0.0001 and gamma =  0.1 is 0.666110183639399
Cross Validation Test Score for C =  0.0001 and gamma =  1 is 0.6661139028475712
Train accuracy for C =  0.0001 and gamma =  1 is 0.666110183639399
Cross Validation Test Score for C =  0.001 and gamma =  0.0001 is 0.6661139028475712
Train accuracy for C =  0.001 and gamma =  0.0001 is 0.666110183639399
Cross Validation Test Score for C =  0.001 and gamma =  0.001 is 0.6661139028475712
Train accuracy for C =  0.001 and gamma =  0.001 is 0.666110183639399
Cross Validation Test Score for C =  0.001 and gamma =  0.01 is 0.6661139028475712
Train accuracy for C =  0.001 and gamma =  0.01 is 0.666110183639399
Cross Validation Test Score for C =  0.001 and gamma =  0.1 is 0.6661139028475712
Train accuracy for C =  0.001 and gamma =  0.1 is 0.666110183639399
Cross Validation Test Score for C =  0.001 and gamma =  1 is 0.7796649916247906
Train accuracy for C =  0.001 and gamma =  1 is 0.7896494156928213
Cross Validation Test Score for C =  0.01 and gamma =  0.0001 is 0.6661139028475712
Train accuracy for C =  0.01 and gamma =  0.0001 is 0.666110183639399
Cross Validation Test Score for C =  0.01 and gamma =  0.001 is 0.6661139028475712
Train accuracy for C =  0.01 and gamma =  0.001 is 0.666110183639399
Cross Validation Test Score for C =  0.01 and gamma =  0.01 is 0.6661139028475712
Train accuracy for C =  0.01 and gamma =  0.01 is 0.666110183639399
Cross Validation Test Score for C =  0.01 and gamma =  0.1 is 0.6661139028475712
Train accuracy for C =  0.01 and gamma =  0.1 is 0.666110183639399
Cross Validation Test Score for C =  0.01 and gamma =  1 is 0.8581993299832495
Train accuracy for C =  0.01 and gamma =  1 is 0.9081803005008348
Cross Validation Test Score for C =  0.1 and gamma =  0.0001 is 0.6661139028475712
Train accuracy for C =  0.1 and gamma =  0.0001 is 0.666110183639399
Cross Validation Test Score for C =  0.1 and gamma =  0.001 is 0.6661139028475712
Train accuracy for C =  0.1 and gamma =  0.001 is 0.666110183639399
Cross Validation Test Score for C =  0.1 and gamma =  0.01 is 0.6661139028475712
Train accuracy for C =  0.1 and gamma =  0.01 is 0.666110183639399
Cross Validation Test Score for C =  0.1 and gamma =  0.1 is 0.6661139028475712
Train accuracy for C =  0.1 and gamma =  0.1 is 0.666110183639399
Cross Validation Test Score for C =  0.1 and gamma =  1 is 0.8598827470686766
Train accuracy for C =  0.1 and gamma =  1 is 0.9048414023372288
Cross Validation Test Score for C =  1 and gamma =  0.0001 is 0.6661139028475712
Train accuracy for C =  1 and gamma =  0.0001 is 0.666110183639399
Cross Validation Test Score for C =  1 and gamma =  0.001 is 0.6661139028475712
Train accuracy for C =  1 and gamma =  0.001 is 0.666110183639399
Cross Validation Test Score for C =  1 and gamma =  0.01 is 0.6661139028475712
Cross Validation Test Score for C =  1 and gamma =  0.01 is 0.6661139028475712
Train accuracy for C =  1 and gamma =  0.01 is 0.666110183639399
Cross Validation Test Score for C =  1 and gamma =  0.1 is 0.7796649916247906
Train accuracy for C =  1 and gamma =  0.1 is 0.7896494156928213
Cross Validation Test Score for C =  1 and gamma =  1 is 0.8832328308207705
Train accuracy for C =  1 and gamma =  1 is 0.8864774624373957
Cross Validation Test Score for C =  10 and gamma =  0.0001 is 0.6661139028475712
Train accuracy for C =  10 and gamma =  0.0001 is 0.666110183639399
Cross Validation Test Score for C =  10 and gamma =  0.001 is 0.6661139028475712
Train accuracy for C =  10 and gamma =  0.001 is 0.666110183639399
Cross Validation Test Score for C =  10 and gamma =  0.01 is 0.6661139028475712
Train accuracy for C =  10 and gamma =  0.01 is 0.666110183639399
Cross Validation Test Score for C =  10 and gamma =  0.1 is 0.8581993299832495
Train accuracy for C =  10 and gamma =  0.1 is 0.9081803005008348
Cross Validation Test Score for C =  10 and gamma =  1 is 0.8865661641541038
Train accuracy for C =  10 and gamma =  1 is 0.8764607679465777
```

In [41]:

```
best_model = SVC(C=10, gamma=1,  kernel='poly').fit(df_2_scaled, df_2_labels)
plot_pairs_svm(best_model, df_2)
```



In [ ]:

```
c = [0.0001, 0.001, 0.01, 0.1, 10]
```

In [ ]:

```
gamma = [0.0001, 0.001, 0.01, 0.1, 1]
```

In [ ]:

```
for i in c:
    for g in gamma:
        clf = OneVsRestClassifier(SVC(C=i, gamma=g,  kernel='poly'))
        cv_results = cross_validate(clf, df2_scaled, df2.label, cv=3)
        print("Cross Validation Test Score for C = ", i, "and gamma = ", g , "is",np.mea
n(cv_results['test_score']))
        train_acc = OneVsRestClassifier(SVC(C=i, gamma=g,  kernel='poly')).fit(df2_scale
d, df2.label).score(df2_scaled, df2.label)
        print("Train accuracy for C = ", i, "and gamma = ", g , "is",train_acc)
```

```
Cross Validation Test Score for C =  0.0001 and gamma =  0.0001 is 0.3589028475711893
Train accuracy for C =  0.0001 and gamma =  0.0001 is 0.35225375626043404
Cross Validation Test Score for C =  0.0001 and gamma =  0.001 is 0.36390284757118924
Train accuracy for C =  0.0001 and gamma =  0.001 is 0.35225375626043404
Cross Validation Test Score for C =  0.0001 and gamma =  0.01 is 0.36390284757118924
Train accuracy for C =  0.0001 and gamma =  0.01 is 0.35225375626043404
Cross Validation Test Score for C =  0.0001 and gamma =  0.1 is 0.36390284757118924
Train accuracy for C =  0.0001 and gamma =  0.1 is 0.35225375626043404
Cross Validation Test Score for C =  0.0001 and gamma =  1 is 0.5676633165829146
Train accuracy for C =  0.0001 and gamma =  1 is 0.5676126878130217
Cross Validation Test Score for C =  0.001 and gamma =  0.0001 is 0.36057788944723623
Train accuracy for C =  0.001 and gamma =  0.0001 is 0.34891485809682804
Cross Validation Test Score for C =  0.001 and gamma =  0.001 is 0.36390284757118924
Cross Validation Test Score for C =  0.001 and gamma =  0.001 is 0.36390284757118924
Train accuracy for C =  0.001 and gamma =  0.001 is 0.35225375626043404
Cross Validation Test Score for C =  0.001 and gamma =  0.01 is 0.36390284757118924
Train accuracy for C =  0.001 and gamma =  0.01 is 0.35225375626043404
Cross Validation Test Score for C =  0.001 and gamma =  0.1 is 0.45073701842546066
Train accuracy for C =  0.001 and gamma =  0.1 is 0.46410684474123537
Cross Validation Test Score for C =  0.001 and gamma =  1 is 0.5826465661641541
Train accuracy for C =  0.001 and gamma =  1 is 0.6026711185308848
Cross Validation Test Score for C =  0.01 and gamma =  0.0001 is 0.36890284757118924
Train accuracy for C =  0.01 and gamma =  0.0001 is 0.34223706176961605
Cross Validation Test Score for C =  0.01 and gamma =  0.001 is 0.36390284757118924
Train accuracy for C =  0.01 and gamma =  0.001 is 0.35225375626043404
Cross Validation Test Score for C =  0.01 and gamma =  0.01 is 0.36390284757118924
Train accuracy for C =  0.01 and gamma =  0.01 is 0.35225375626043404
Cross Validation Test Score for C =  0.01 and gamma =  0.1 is 0.5576130653266332
Train accuracy for C =  0.01 and gamma =  0.1 is 0.5859766277128547
Cross Validation Test Score for C =  0.01 and gamma =  1 is 0.627713567839196
Train accuracy for C =  0.01 and gamma =  1 is 0.6410684474123539
Cross Validation Test Score for C =  0.1 and gamma =  0.0001 is 0.36390284757118924
Train accuracy for C =  0.1 and gamma =  0.0001 is 0.35225375626043404
Cross Validation Test Score for C =  0.1 and gamma =  0.001 is 0.36390284757118924
Train accuracy for C =  0.1 and gamma =  0.001 is 0.35225375626043404
Cross Validation Test Score for C =  0.1 and gamma =  0.01 is 0.36390284757118924
Train accuracy for C =  0.1 and gamma =  0.01 is 0.35225375626043404
Cross Validation Test Score for C =  0.1 and gamma =  0.1 is 0.565996649916248
Train accuracy for C =  0.1 and gamma =  0.1 is 0.5692821368948247
Cross Validation Test Score for C =  0.1 and gamma =  1 is 0.6361557788944724
Train accuracy for C =  0.1 and gamma =  1 is 0.669449081803005
Cross Validation Test Score for C =  10 and gamma =  0.0001 is 0.36390284757118924
Train accuracy for C =  10 and gamma =  0.0001 is 0.35225375626043404
Cross Validation Test Score for C =  10 and gamma =  0.001 is 0.36390284757118924
Train accuracy for C =  10 and gamma =  0.001 is 0.35225375626043404
Cross Validation Test Score for C =  10 and gamma =  0.01 is 0.5576130653266332
Train accuracy for C =  10 and gamma =  0.01 is 0.5859766277128547
Cross Validation Test Score for C =  10 and gamma =  0.1 is 0.627713567839196
Train accuracy for C =  10 and gamma =  0.1 is 0.6410684474123539
Cross Validation Test Score for C =  10 and gamma =  1 is 0.6729396984924622
Train accuracy for C =  10 and gamma =  1 is 0.6928213689482471
```

In [ ]:

```
best_model  = OneVsRestClassifier(SVC(C=10, gamma=1,  kernel='poly')).fit(df2_scaled, df
2.label)
```

In [ ]:

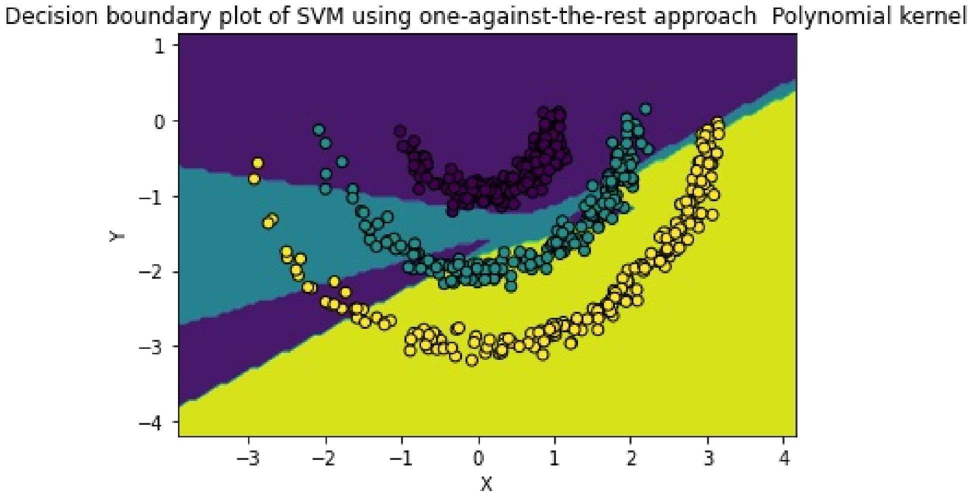
```
X = df2[['x','y']].values
```

In [ ]:

```
xmin, xmax = np.min(X[:,0]) - 1, np.max(X[:,0]) + 1
ymin, ymax = np.min(X[:,1]) - 1, np.max(X[:,1]) + 1

x = np.linspace(xmin,xmax,100)
y = np.linspace(ymin,ymax,100)

xx, yy = np.meshgrid(x,y)

Z = np.stack((xx.ravel(),yy.ravel()),axis = 1)

Z = scaler.transform(Z)
pred = best_model.predict(Z).reshape(100,100)
# print(pred.shape)

plt.contourf(xx,yy,pred)
plt.scatter(X[:,0],X[:,1],c = df2.label, edgecolors= 'k')
plt.xlabel("X")
plt.ylabel("Y")
plt.title("Decision boundary plot of SVM using one-against-the-rest approach  Polynomial
kernel")
```

Out[ ]:

```
Text(0.5, 1.0, 'Decision boundary plot of SVM using one-against-the-rest approach  Polyno
mial kernel')
```


Decision boundary plot of SVM using one-against-the-rest approach Polynomial kernel

In [ ]:

```
confusion_matrix(df2.label, best_model.predict(df2_scaled))
```

Out[ ]:

```
array([[198,   1,   0],
       [ 67,  42,  91],
       [ 12,  13, 175]])
```

In [ ]:

```
confusion_matrix(df2_test.label, best_model.predict(df2_test_scaled))
```

Out[ ]:

```
array([[29,  0,  0],
       [13,  6, 11],
       [ 1,  3, 26]])
```

In [ ]:

```
test_acc = best_model.score(df2_test_scaled, df2_test.label)
print("Test accuracy for C = ", 10, "and gamma = ", 1 , "is",test_acc)
```

```
Test accuracy for C =  10 and gamma =  1 is 0.6853932584269663
```

DATASET 2

In [ ]:

```
datasets = []
```

In [ ]:

```
classes = ["coast", "highway", "insidecity","opencountry", "tallbuilding"]
```

In [ ]:

```
for i in classes:
  df = pd.read_csv("/content/drive/MyDrive/prml_assign/Dataset_2A_12/" + i + "/train.csv
")
  datasets.append(df.drop(columns=["image_names"]).values)
```

In [ ]:

```
all_data = []
```

In [ ]:

```
labels = []
```

In [ ]:

```
for index, i in enumerate(datasets):
    for j in i:
        all_data.append(j)
        labels.append(index)
```

In [ ]:

```
all_data = np.array(all_data)
```

In [ ]:

```
datasets_test = []
```

In [ ]:

```
classes = ["coast", "highway", "insidecity","opencountry", "tallbuilding"]
```

In [ ]:

```
for i in classes:
  df = pd.read_csv("/content/drive/MyDrive/prml_assign/Dataset_2A_12/" + i + "/dev.csv")
  datasets_test.append(df.drop(columns=["image_names"]).values)
```

In [ ]:

```
all_data_test = []
```

In [ ]:

```
labels_test = []
```

In [ ]:

```
for index, i in enumerate(datasets_test):
    for j in i:
        all_data_test.append(j)
        labels_test.append(index)
```

In [ ]:

```
all_data_test = np.array(all_data_test)
```

In [ ]:

```
all_data_test.shape
```

Out[ ]:

```
(340, 24)
```

In [ ]:

```
scaler = StandardScaler().fit(all_data)
```

In [ ]:

```
all_data_scaled = scaler.transform(all_data)
```

In [ ]:

```
all_data_test_scaled = scaler.transform(all_data_test)
```

In [ ]:

```
all_data_scaled.shape
```

Out[ ]:

```
(1184, 24)
```

In [ ]:

```
all_data_test_scaled.shape
```

Out[ ]:

```
(340, 24)
```

MLFFNN

In [ ]:

```
alpha = [0.00001, 0.0001, 0.001, 0.01, 0.1]
```

In [ ]:

```
hidden_layer_size_1 = [50, 100, 150]
hidden_layer_size_2 = [50, 100, 150]
```

In [ ]:

```
for i in alpha:
    for j in hidden_layer_size_1:
        for k in hidden_layer_size_2:
            clf = MLPClassifier(solver='lbfgs', alpha=i,hidden_layer_sizes=(j, k), rando
m_state=42)
            cv_results = cross_validate(clf, all_data_scaled, labels, cv=3)
            print("Cross Validation Test Score for learning rate = ", i, "and number of
hidden units layer 1 = ", j ,"and number of hidden units layer 2 = ", k , "is",np.mean(c
v_results['test_score']))
            train_acc = MLPClassifier(solver='lbfgs', alpha=i,hidden_layer_sizes=(j, k),
random_state=42).fit(all_data_scaled, labels).score(all_data_scaled, labels)
            print("Train accuracy for learning rate = ", i, "and number of hidden units
layer 1 = ", j ,"and number of hidden units layer 2 = ", k , "is",train_acc)
```

```
Cross Validation Test Score for learning rate =  1e-05 and number of hidden units layer 1
=  50 and number of hidden units layer 2 =  50 is 0.5861294951701687
Train accuracy for learning rate =  1e-05 and number of hidden units layer 1 =  50 and nu
mber of hidden units layer 2 =  50 is 1.0
Cross Validation Test Score for learning rate =  1e-05 and number of hidden units layer 1
=  50 and number of hidden units layer 2 =  100 is 0.5734691254899441
Train accuracy for learning rate =  1e-05 and number of hidden units layer 1 =  50 and nu
mber of hidden units layer 2 =  100 is 1.0
Cross Validation Test Score for learning rate =  1e-05 and number of hidden units layer 1
=  50 and number of hidden units layer 2 =  150 is 0.5667116451412538
Train accuracy for learning rate =  1e-05 and number of hidden units layer 1 =  50 and nu
mber of hidden units layer 2 =  150 is 1.0
Cross Validation Test Score for learning rate =  1e-05 and number of hidden units layer 1
=  100 and number of hidden units layer 2 =  50 is 0.5785217074685686
Train accuracy for learning rate =  1e-05 and number of hidden units layer 1 =  100 and n
umber of hidden units layer 2 =  50 is 1.0
Cross Validation Test Score for learning rate =  1e-05 and number of hidden units layer 1
=  100 and number of hidden units layer 2 =  100 is 0.5928934010152284
Train accuracy for learning rate =  1e-05 and number of hidden units layer 1 =  100 and n
umber of hidden units layer 2 =  100 is 1.0
Cross Validation Test Score for learning rate =  1e-05 and number of hidden units layer 1
=  100 and number of hidden units layer 2 =  150 is 0.5912206301270106
Train accuracy for learning rate =  1e-05 and number of hidden units layer 1 =  100 and n
umber of hidden units layer 2 =  150 is 1.0
Cross Validation Test Score for learning rate =  1e-05 and number of hidden units layer 1
=  150 and number of hidden units layer 2 =  50 is 0.5937372828717685
Train accuracy for learning rate =  1e-05 and number of hidden units layer 1 =  150 and n
umber of hidden units layer 2 =  50 is 1.0
Cross Validation Test Score for learning rate =  1e-05 and number of hidden units layer 1
=  150 and number of hidden units layer 2 =  100 is 0.5920538028229347
Train accuracy for learning rate =  1e-05 and number of hidden units layer 1 =  150 and n
```

```
umber of hidden units layer 2 =  100 is 1.0
Cross Validation Test Score for learning rate =  1e-05 and number of hidden units layer 1
=  150 and number of hidden units layer 2 =  150 is 0.5895114480926985
Train accuracy for learning rate =  1e-05 and number of hidden units layer 1 =  150 and n
umber of hidden units layer 2 =  150 is 1.0
Cross Validation Test Score for learning rate =  0.0001 and number of hidden units layer
1 =  50 and number of hidden units layer 2 =  50 is 0.5810640621988049
Train accuracy for learning rate =  0.0001 and number of hidden units layer 1 =  50 and n
umber of hidden units layer 2 =  50 is 1.0
Cross Validation Test Score for learning rate =  0.0001 and number of hidden units layer
1 =  50 and number of hidden units layer 2 =  100 is 0.5759900618989483
Train accuracy for learning rate =  0.0001 and number of hidden units layer 1 =  50 and n
umber of hidden units layer 2 =  100 is 1.0
Cross Validation Test Score for learning rate =  0.0001 and number of hidden units layer
1 =  50 and number of hidden units layer 2 =  150 is 0.5608023303133501
Train accuracy for learning rate =  0.0001 and number of hidden units layer 1 =  50 and n
umber of hidden units layer 2 =  150 is 1.0
Cross Validation Test Score for learning rate =  0.0001 and number of hidden units layer
1 =  100 and number of hidden units layer 2 =  50 is 0.5810512112060656
Train accuracy for learning rate =  0.0001 and number of hidden units layer 1 =  100 and
number of hidden units layer 2 =  50 is 1.0
Cross Validation Test Score for learning rate =  0.0001 and number of hidden units layer
1 =  100 and number of hidden units layer 2 =  100 is 0.5979588339865921
Train accuracy for learning rate =  0.0001 and number of hidden units layer 1 =  100 and
number of hidden units layer 2 =  100 is 1.0
Cross Validation Test Score for learning rate =  0.0001 and number of hidden units layer
1 =  100 and number of hidden units layer 2 =  150 is 0.5920623701514276
Train accuracy for learning rate =  0.0001 and number of hidden units layer 1 =  100 and
number of hidden units layer 2 =  150 is 1.0
Cross Validation Test Score for learning rate =  0.0001 and number of hidden units layer
1 =  150 and number of hidden units layer 2 =  50 is 0.5878108333868791
Train accuracy for learning rate =  0.0001 and number of hidden units layer 1 =  150 and
number of hidden units layer 2 =  50 is 1.0
Cross Validation Test Score for learning rate =  0.0001 and number of hidden units layer
1 =  150 and number of hidden units layer 2 =  100 is 0.5920623701514276
Train accuracy for learning rate =  0.0001 and number of hidden units layer 1 =  150 and
number of hidden units layer 2 =  100 is 1.0
Cross Validation Test Score for learning rate =  0.0001 and number of hidden units layer
1 =  150 and number of hidden units layer 2 =  150 is 0.5962646447771424
Train accuracy for learning rate =  0.0001 and number of hidden units layer 1 =  150 and
number of hidden units layer 2 =  150 is 1.0
Cross Validation Test Score for learning rate =  0.001 and number of hidden units layer 1
=  50 and number of hidden units layer 2 =  50 is 0.5928805500224893
Train accuracy for learning rate =  0.001 and number of hidden units layer 1 =  50 and nu
mber of hidden units layer 2 =  50 is 1.0
Cross Validation Test Score for learning rate =  0.001 and number of hidden units layer 1
=  50 and number of hidden units layer 2 =  100 is 0.5852898969778749
Train accuracy for learning rate =  0.001 and number of hidden units layer 1 =  50 and nu
mber of hidden units layer 2 =  100 is 1.0
Cross Validation Test Score for learning rate =  0.001 and number of hidden units layer 1
=  50 and number of hidden units layer 2 =  150 is 0.5607980466491037
Train accuracy for learning rate =  0.001 and number of hidden units layer 1 =  50 and nu
mber of hidden units layer 2 =  150 is 1.0
Cross Validation Test Score for learning rate =  0.001 and number of hidden units layer 1
=  100 and number of hidden units layer 2 =  50 is 0.5751504637066547
Train accuracy for learning rate =  0.001 and number of hidden units layer 1 =  100 and n
umber of hidden units layer 2 =  50 is 1.0
Cross Validation Test Score for learning rate =  0.001 and number of hidden units layer 1
=  100 and number of hidden units layer 2 =  100 is 0.5988027158431323
Train accuracy for learning rate =  0.001 and number of hidden units layer 1 =  100 and n
umber of hidden units layer 2 =  100 is 1.0
Cross Validation Test Score for learning rate =  0.001 and number of hidden units layer 1
=  100 and number of hidden units layer 2 =  150 is 0.5852920388099981
Train accuracy for learning rate =  0.001 and number of hidden units layer 1 =  100 and n
umber of hidden units layer 2 =  150 is 1.0
Cross Validation Test Score for learning rate =  0.001 and number of hidden units layer 1
=  150 and number of hidden units layer 2 =  50 is 0.5928762663582429
Train accuracy for learning rate =  0.001 and number of hidden units layer 1 =  150 and n
umber of hidden units layer 2 =  50 is 1.0
Cross Validation Test Score for learning rate =  0.001 and number of hidden units layer 1
=  150 and number of hidden units layer 2 =  100 is 0.5920623701514276
Train accuracy for learning rate =  0.001 and number of hidden units layer 1 =  150 and n
```

```
umber of hidden units layer 2 =  100 is 1.0
Cross Validation Test Score for learning rate =  0.001 and number of hidden units layer 1
=  150 and number of hidden units layer 2 =  150 is 0.5979566921544689
Train accuracy for learning rate =  0.001 and number of hidden units layer 1 =  150 and n
umber of hidden units layer 2 =  150 is 1.0
Cross Validation Test Score for learning rate =  0.01 and number of hidden units layer 1
=  50 and number of hidden units layer 2 =  50 is 0.597104242969436
Train accuracy for learning rate =  0.01 and number of hidden units layer 1 =  50 and num
ber of hidden units layer 2 =  50 is 1.0
Cross Validation Test Score for learning rate =  0.01 and number of hidden units layer 1
=  50 and number of hidden units layer 2 =  100 is 0.5895093062605753
Train accuracy for learning rate =  0.01 and number of hidden units layer 1 =  50 and num
ber of hidden units layer 2 =  100 is 1.0
Cross Validation Test Score for learning rate =  0.01 and number of hidden units layer 1
=  50 and number of hidden units layer 2 =  150 is 0.5802201803422647
Train accuracy for learning rate =  0.01 and number of hidden units layer 1 =  50 and num
ber of hidden units layer 2 =  150 is 1.0
Cross Validation Test Score for learning rate =  0.01 and number of hidden units layer 1
=  100 and number of hidden units layer 2 =  50 is 0.5945790228961854
Train accuracy for learning rate =  0.01 and number of hidden units layer 1 =  100 and nu
mber of hidden units layer 2 =  50 is 1.0
Cross Validation Test Score for learning rate =  0.01 and number of hidden units layer 1
=  100 and number of hidden units layer 2 =  100 is 0.6080811326008267
Train accuracy for learning rate =  0.01 and number of hidden units layer 1 =  100 and nu
mber of hidden units layer 2 =  100 is 1.0
Cross Validation Test Score for learning rate =  0.01 and number of hidden units layer 1
=  100 and number of hidden units layer 2 =  150 is 0.601351496069738
Train accuracy for learning rate =  0.01 and number of hidden units layer 1 =  100 and nu
mber of hidden units layer 2 =  150 is 1.0
Cross Validation Test Score for learning rate =  0.01 and number of hidden units layer 1
=  150 and number of hidden units layer 2 =  50 is 0.6089292981216132
Train accuracy for learning rate =  0.01 and number of hidden units layer 1 =  150 and nu
mber of hidden units layer 2 =  50 is 1.0
Cross Validation Test Score for learning rate =  0.01 and number of hidden units layer 1
=  150 and number of hidden units layer 2 =  100 is 0.6114695110197262
Train accuracy for learning rate =  0.01 and number of hidden units layer 1 =  150 and nu
mber of hidden units layer 2 =  100 is 1.0
Cross Validation Test Score for learning rate =  0.01 and number of hidden units layer 1
=  150 and number of hidden units layer 2 =  150 is 0.6072479599049027
Train accuracy for learning rate =  0.01 and number of hidden units layer 1 =  150 and nu
mber of hidden units layer 2 =  150 is 1.0
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:4
70: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
  self.n_iter_ = _check_optimize_result("lbfgs", opt_res, self.max_iter)
/usr/local/lib/python3.7/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:4
70: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
  self.n_iter_ = _check_optimize_result("lbfgs", opt_res, self.max_iter)
/usr/local/lib/python3.7/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:4
70: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
  self.n_iter_ = _check_optimize_result("lbfgs", opt_res, self.max_iter)
```

```
Cross Validation Test Score for learning rate =  0.1 and number of hidden units layer 1 =
50 and number of hidden units layer 2 =  50 is 0.610623487331063
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:4
70: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
  self.n_iter_ = _check_optimize_result("lbfgs", opt_res, self.max_iter)
```

```
Train accuracy for learning rate =  0.1 and number of hidden units layer 1 =  50 and numb
er of hidden units layer 2 =  50 is 1.0
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:4
70: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
  self.n_iter_ = _check_optimize_result("lbfgs", opt_res, self.max_iter)
/usr/local/lib/python3.7/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:4
70: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
  self.n_iter_ = _check_optimize_result("lbfgs", opt_res, self.max_iter)
/usr/local/lib/python3.7/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:4
70: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
  self.n_iter_ = _check_optimize_result("lbfgs", opt_res, self.max_iter)
```

```
Cross Validation Test Score for learning rate =  0.1 and number of hidden units layer 1 =
50 and number of hidden units layer 2 =  100 is 0.6123069673798968
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:4
70: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
  self.n_iter_ = _check_optimize_result("lbfgs", opt_res, self.max_iter)
```

```
Train accuracy for learning rate =  0.1 and number of hidden units layer 1 =  50 and numb
er of hidden units layer 2 =  100 is 1.0
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:4
70: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
  self.n_iter_ = _check_optimize_result("lbfgs", opt_res, self.max_iter)
/usr/local/lib/python3.7/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:4
70: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
  self.n_iter_ = _check_optimize_result("lbfgs", opt_res, self.max_iter)
/usr/local/lib/python3.7/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:4
70: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
  self.n_iter_ = _check_optimize_result("lbfgs", opt_res, self.max_iter)
```

```
Cross Validation Test Score for learning rate =  0.1 and number of hidden units layer 1 =
50 and number of hidden units layer 2 =  150 is 0.5988027158431322
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:4
70: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
  self.n_iter_ = _check_optimize_result("lbfgs", opt_res, self.max_iter)
```

```
Train accuracy for learning rate =  0.1 and number of hidden units layer 1 =  50 and numb
er of hidden units layer 2 =  150 is 1.0
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:4
70: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
  self.n_iter_ = _check_optimize_result("lbfgs", opt_res, self.max_iter)
/usr/local/lib/python3.7/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:4
70: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
  self.n_iter_ = _check_optimize_result("lbfgs", opt_res, self.max_iter)
/usr/local/lib/python3.7/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:4
70: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
  self.n_iter_ = _check_optimize_result("lbfgs", opt_res, self.max_iter)
```

```
Cross Validation Test Score for learning rate =  0.1 and number of hidden units layer 1 =
100 and number of hidden units layer 2 =  50 is 0.6047163143352824
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:4
70: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
  self.n_iter_ = _check_optimize_result("lbfgs", opt_res, self.max_iter)
```

```
Train accuracy for learning rate =  0.1 and number of hidden units layer 1 =  100 and num
ber of hidden units layer 2 =  50 is 1.0
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:4
70: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
  self.n_iter_ = _check_optimize_result("lbfgs", opt_res, self.max_iter)
/usr/local/lib/python3.7/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:4
70: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
  self.n_iter_ = _check_optimize_result("lbfgs", opt_res, self.max_iter)
/usr/local/lib/python3.7/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:4
70: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
  self.n_iter_ = _check_optimize_result("lbfgs", opt_res, self.max_iter)
```

```
Cross Validation Test Score for learning rate =  0.1 and number of hidden units layer 1 =
100 and number of hidden units layer 2 =  100 is 0.6114695110197262
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:4
70: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
  self.n_iter_ = _check_optimize_result("lbfgs", opt_res, self.max_iter)
```

```
Train accuracy for learning rate =  0.1 and number of hidden units layer 1 =  100 and num
```

```
ber of hidden units layer 2 =  100 is 1.0
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:4
70: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
  self.n_iter_ = _check_optimize_result("lbfgs", opt_res, self.max_iter)
/usr/local/lib/python3.7/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:4
70: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
  self.n_iter_ = _check_optimize_result("lbfgs", opt_res, self.max_iter)
/usr/local/lib/python3.7/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:4
70: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
  self.n_iter_ = _check_optimize_result("lbfgs", opt_res, self.max_iter)
```

```
Cross Validation Test Score for learning rate =  0.1 and number of hidden units layer 1 =
100 and number of hidden units layer 2 =  150 is 0.6190837242176958
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:4
70: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
  self.n_iter_ = _check_optimize_result("lbfgs", opt_res, self.max_iter)
```

```
Train accuracy for learning rate =  0.1 and number of hidden units layer 1 =  100 and num
ber of hidden units layer 2 =  150 is 1.0
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:4
70: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
  self.n_iter_ = _check_optimize_result("lbfgs", opt_res, self.max_iter)
/usr/local/lib/python3.7/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:4
70: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
  self.n_iter_ = _check_optimize_result("lbfgs", opt_res, self.max_iter)
/usr/local/lib/python3.7/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:4
70: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
  self.n_iter_ = _check_optimize_result("lbfgs", opt_res, self.max_iter)
```

```
Cross Validation Test Score for learning rate =  0.1 and number of hidden units layer 1 =
150 and number of hidden units layer 2 =  50 is 0.6165285184947203
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:4
70: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
  self.n_iter_ = _check_optimize_result("lbfgs", opt_res, self.max_iter)
```

```
Train accuracy for learning rate =  0.1 and number of hidden units layer 1 =  150 and num
ber of hidden units layer 2 =  50 is 1.0
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:4
70: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
  self.n_iter_ = _check_optimize_result("lbfgs", opt_res, self.max_iter)
/usr/local/lib/python3.7/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:4
70: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
  self.n_iter_ = _check_optimize_result("lbfgs", opt_res, self.max_iter)
/usr/local/lib/python3.7/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:4
70: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
  self.n_iter_ = _check_optimize_result("lbfgs", opt_res, self.max_iter)
```

```
Cross Validation Test Score for learning rate =  0.1 and number of hidden units layer 1 =
150 and number of hidden units layer 2 =  100 is 0.6283749919681295
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:4
70: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
  self.n_iter_ = _check_optimize_result("lbfgs", opt_res, self.max_iter)
```

```
Train accuracy for learning rate =  0.1 and number of hidden units layer 1 =  150 and num
ber of hidden units layer 2 =  100 is 1.0
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:4
70: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
  self.n_iter_ = _check_optimize_result("lbfgs", opt_res, self.max_iter)
/usr/local/lib/python3.7/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:4
70: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
  self.n_iter_ = _check_optimize_result("lbfgs", opt_res, self.max_iter)
/usr/local/lib/python3.7/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:4
70: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
  self.n_iter_ = _check_optimize_result("lbfgs", opt_res, self.max_iter)
```

```
Cross Validation Test Score for learning rate =  0.1 and number of hidden units layer 1 =
150 and number of hidden units layer 2 =  150 is 0.6097796054745229
Train accuracy for learning rate =  0.1 and number of hidden units layer 1 =  150 and num
ber of hidden units layer 2 =  150 is 1.0
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:4
70: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
  self.n_iter_ = _check_optimize_result("lbfgs", opt_res, self.max_iter)
```

```
In [ ]:
```

```
best_model  = MLPClassifier(solver='lbfgs', alpha=0.1,hidden_layer_sizes=(150, 100), rand
om_state=42).fit(all_data_scaled, labels)
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:4
70: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
  self.n_iter_ = _check_optimize_result("lbfgs", opt_res, self.max_iter)
```

In [ ]:

```
confusion_matrix(labels, best_model.predict(all_data_scaled))
```

Out[ ]:

```
array([[251,   0,   0,   0,   0],
       [  0, 182,   0,   0,   0],
       [  0,   0, 215,   0,   0],
       [  0,   0,   0, 287,   0],
       [  0,   0,   0,   0, 249]])
```

In [ ]:

```
confusion_matrix(labels_test, best_model.predict(all_data_test_scaled))
```

Out[ ]:

```
array([[40,  7,  9,  9,  8],
       [ 5, 24,  5,  8, 10],
       [ 1,  1, 42, 10,  8],
       [ 9,  2,  9, 53,  9],
       [19,  3, 13,  6, 30]])
```

In [ ]:

```
test_acc = best_model.score(all_data_test_scaled, labels_test)
print("Test accuracy for alpha = ", 0.1, "and number of hidden units layer 1 = ", 150 ,"
and number of hidden units layer 2 = ", 100 , "is", test_acc)
```

```
Test accuracy for alpha =  0.1 and number of hidden units layer 1 =  150 and number of hi
dden units layer 2 =  100 is 0.5558823529411765
```

SVM - One v/s rest

In [ ]:

```
c = [0.0001, 0.001, 0.01, 0.1, 1]
```

In [ ]:

```
gamma = [0.0001, 0.001, 0.01, 0.1, 1]
```

In [ ]:

```
for i in c:
    for g in gamma:
        clf = OneVsRestClassifier(SVC(C=i, gamma=g,  kernel='rbf'))
        cv_results = cross_validate(clf, all_data_scaled, labels, cv=3)
        print("Cross Validation Test Score for C = ", i, "and gamma = ", g , "is",np.mea
n(cv_results['test_score']))
        train_acc = OneVsRestClassifier(SVC(C=i, gamma=g,  kernel='rbf')).fit(all_data_s
caled, labels).score(all_data_scaled, labels)
        print("Train accuracy for C = ", i, "and gamma = ", g , "is",train_acc)
```

```
Cross Validation Test Score for C =  0.0001 and gamma =  0.0001 is 0.35218359784960057
Train accuracy for C =  0.0001 and gamma =  0.0001 is 0.36908783783783783
Cross Validation Test Score for C =  0.0001 and gamma =  0.001 is 0.40877080254449655
Train accuracy for C =  0.0001 and gamma =  0.001 is 0.39780405405405406
```

```
Cross Validation Test Score for C =  0.0001 and gamma =  0.01 is 0.4746428494934567
Train accuracy for C =  0.0001 and gamma =  0.01 is 0.5304054054054054
Cross Validation Test Score for C =  0.0001 and gamma =  0.1 is 0.5666966523163914
Train accuracy for C =  0.0001 and gamma =  0.1 is 0.7981418918918919
Cross Validation Test Score for C =  0.0001 and gamma =  1 is 0.33537021568249487
Train accuracy for C =  0.0001 and gamma =  1 is 1.0
Cross Validation Test Score for C =  0.001 and gamma =  0.0001 is 0.35218359784960057
Train accuracy for C =  0.001 and gamma =  0.0001 is 0.36908783783783783
Cross Validation Test Score for C =  0.001 and gamma =  0.001 is 0.40877080254449655
Train accuracy for C =  0.001 and gamma =  0.001 is 0.39780405405405406
Cross Validation Test Score for C =  0.001 and gamma =  0.01 is 0.5092869840861874
Train accuracy for C =  0.001 and gamma =  0.01 is 0.5929054054054054
Cross Validation Test Score for C =  0.001 and gamma =  0.1 is 0.5979545503223457
Train accuracy for C =  0.001 and gamma =  0.1 is 0.8505067567567568
Cross Validation Test Score for C =  0.001 and gamma =  1 is 0.41304589946240017
Train accuracy for C =  0.001 and gamma =  1 is 1.0
Cross Validation Test Score for C =  0.01 and gamma =  0.0001 is 0.35218359784960057
Train accuracy for C =  0.01 and gamma =  0.0001 is 0.38091216216216217
Cross Validation Test Score for C =  0.01 and gamma =  0.001 is 0.47717021139883053
Train accuracy for C =  0.01 and gamma =  0.001 is 0.5211148648648649
Cross Validation Test Score for C =  0.01 and gamma =  0.01 is 0.5371457945126261
Train accuracy for C =  0.01 and gamma =  0.01 is 0.5878378378378378
Cross Validation Test Score for C =  0.01 and gamma =  0.1 is 0.599644455867549
Train accuracy for C =  0.01 and gamma =  0.1 is 0.8547297297297297
Cross Validation Test Score for C =  0.01 and gamma =  1 is 0.3555698344363769
Train accuracy for C =  0.01 and gamma =  1 is 1.0
Cross Validation Test Score for C =  0.1 and gamma =  0.0001 is 0.4662083145923022
Train accuracy for C =  0.1 and gamma =  0.0001 is 0.4780405405405405
Cross Validation Test Score for C =  0.1 and gamma =  0.001 is 0.4805650153140997
Train accuracy for C =  0.1 and gamma =  0.001 is 0.5228040540540541
Cross Validation Test Score for C =  0.1 and gamma =  0.01 is 0.5362954871597164
Train accuracy for C =  0.1 and gamma =  0.01 is 0.5912162162162162
Cross Validation Test Score for C =  0.1 and gamma =  0.1 is 0.6004904795562124
Train accuracy for C =  0.1 and gamma =  0.1 is 0.8572635135135135
Cross Validation Test Score for C =  0.1 and gamma =  1 is 0.356413716292917
Train accuracy for C =  0.1 and gamma =  1 is 1.0
Cross Validation Test Score for C =  1 and gamma =  0.0001 is 0.45859624322645587
Train accuracy for C =  1 and gamma =  0.0001 is 0.4839527027027027
Cross Validation Test Score for C =  1 and gamma =  0.001 is 0.47549744051061277
Train accuracy for C =  1 and gamma =  0.001 is 0.5194256756756757
Cross Validation Test Score for C =  1 and gamma =  0.01 is 0.5861402043307846
Train accuracy for C =  1 and gamma =  0.01 is 0.6621621621621622
Cross Validation Test Score for C =  1 and gamma =  0.1 is 0.6249844717171068
Train accuracy for C =  1 and gamma =  0.1 is 0.9307432432432432
Cross Validation Test Score for C =  1 and gamma =  1 is 0.3547238107477136
Train accuracy for C =  1 and gamma =  1 is 1.0
```

In [ ]:

```
best_model  = OneVsRestClassifier(SVC(C=1, gamma=0.1,  kernel='rbf')).fit(all_data_scaled, labels)
```

In [ ]:

```
confusion_matrix(labels, best_model.predict(all_data_scaled))
```

Out[ ]:

```
array([[227,   0,  10,   4,  10],
       [  0, 175,   3,   3,   1],
       [  0,   0, 198,   5,  12],
       [  3,   0,   8, 271,   5],
       [  2,   0,  12,   4, 231]])
```

In [ ]:

```
confusion_matrix(labels_test, best_model.predict(all_data_test_scaled))
```

Out[ ]:

```
array([[42,  8,  8,  8,  7],
       [ 9, 24,  5,  6,  8],
       [ 4,  0, 41,  8,  9],
       [ 7,   1, 12, 54,   8],
       [ 8,   3, 17, 14, 29]])
```

In [ ]:

```
test_acc = best_model.score(all_data_test_scaled, labels_test)
print("Train accuracy for C = ", 1, "and gamma = ", 0.1 , "is",test_acc)
```

```
Train accuracy for C =  1 and gamma =  0.1 is 0.5588235294117647
```

Plotting the activation surfaces for the neural network for dataset 2B, for the best configuration.

In [ ]:

```
from numpy import loadtxt
from keras.models import Sequential
from keras.layers import Dense
import pandas as pd
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
from sklearn.preprocessing import LabelEncoder
from keras.utils import np_utils
import math
```

In [ ]:

```
from numpy.random import seed
seed(1)
```

In [ ]:

```
df = pd.read_csv('/content/train (1).csv')
```

In [ ]:

```
df.columns = ['x1', 'x2', 'y']
```

In [ ]:

```
df.head()
```

Out[ ]:

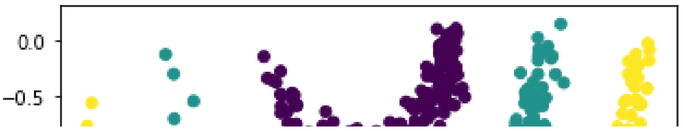
	x1	x2	y
0	0.963906	-0.084686	0.0
1	0.751577	-0.692578	0.0
2	0.064676	-1.029066	0.0
3	0.945798	-0.525876	0.0
4	1.057655	0.000716	0.0

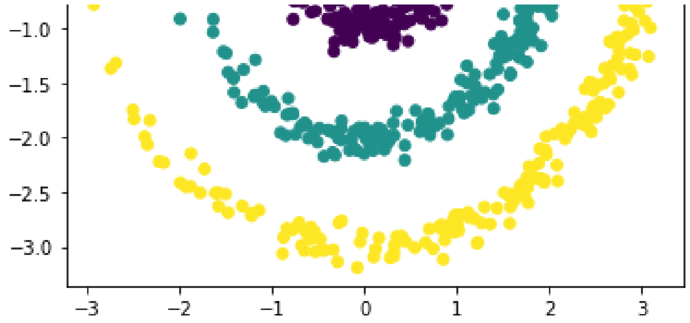
In [ ]:

```
plt.scatter(df.x1, df.x2, c = df.y)
```

Out[ ]:

```
<matplotlib.collections.PathCollection at 0x7f42e22d4750>
```





In [ ]:

```
X = df.drop(columns= ["y"]).values
```

In [ ]:

```
X.shape
```

Out[ ]:

```
(599, 2)
```

In [ ]:

```
y.shape
```

Out[ ]:

```
(599,)
```

In [ ]:

```
y = df.y.values
```

In [ ]:

```
encoder = LabelEncoder()
encoder.fit(y)
encoded_Y = encoder.transform(y)
# convert integers to dummy variables (i.e. one hot encoded)
dummy_y = np_utils.to_categorical(encoded_Y)
```

In [ ]:

```
dummy_y
```

Out[ ]:

```
array([[1., 0., 0.],
       [1., 0., 0.],
       [1., 0., 0.],
       ...,
       [0., 0., 1.],
       [0., 0., 1.],
       [0., 0., 1.]], dtype=float32)
```

In [ ]:

```
# custom function
def sigmoid(x):
  return 1 / (1 + math.exp(-x))

# define vectorized sigmoid
sigmoid_v = np.vectorize(sigmoid)
```

In [ ]:

```
def find_activation_layer_1(x, y, node_number,  model):
    layers = model.layers
    weight_matrix = layers[0].get_weights()[0]
    bias = layers[0].get_weights()[1]
    point = np.array([x, y]).T
    activations = weight_matrix.T@point + bias
    return max(0, activations[node_number])
```

In [ ]:

```
def find_activation_layer_2(x, y, node_number,  model):
    layers = model.layers
    weight_matrix_0 = layers[0].get_weights()[0]
    bias_0 = layers[0].get_weights()[1]
    weight_matrix_1 = layers[1].get_weights()[0]
    bias_1 = layers[1].get_weights()[1]

    point = np.array([x, y]).T
    activations_0 = np.maximum(weight_matrix_0.T@point + bias_0, 0)
    activations_1 = np.maximum(weight_matrix_1.T@activations_0 + bias_1, 0)
    return max(0, activations_1[node_number])
```

In [ ]:

```
def find_activation_layer_3(x, y, node_number,  model):
    layers = model.layers
    weight_matrix_0 = layers[0].get_weights()[0]
    bias_0 = layers[0].get_weights()[1]
    weight_matrix_1 = layers[1].get_weights()[0]
    bias_1 = layers[1].get_weights()[1]
    weight_matrix_2 = layers[2].get_weights()[0]
    bias_2 = layers[2].get_weights()[1]


    point = np.array([x, y]).T
    activations_0 = np.maximum(weight_matrix_0.T@point + bias_0, 0)
    activations_1 = np.maximum(weight_matrix_1.T@activations_0 + bias_1, 0)
    activations_2 = weight_matrix_2.T@activations_1 + bias_2
    return sigmoid_v(activations_2[node_number])
    # return  model.predict(np.array([x,y]).reshape(1,2))
```

In [ ]:

```
def find_activation(x, y, layer_number, node_number, model):
    if(layer_number == 0):
        return find_activation_layer_1(x, y, node_number,  model)
    elif(layer_number == 1):
        return find_activation_layer_2(x, y, node_number,  model)
    else:
        return find_activation_layer_3(x, y, node_number,  model)
```

In [ ]:

```
a = np.linspace(-3, 3, 100)
b = np.linspace(-3, 0, 100)

A, B = np.meshgrid(a, b)

print(A.shape, B.shape)
```

```
(100, 100) (100, 100)
```

In [ ]:

```
def plot_activation(model, layer_number):
    for m in range(3):
        Z = np.zeros((100, 100))
        for i in range(100):
            for j in range(100):
                Z[i][j] = find_activation(A[i][j], B[i][j], layer_number, m, model)
        fig = plt.figure()
        axes = fig.gca(projection ='3d')
        axes.plot_surface(A, B, Z)
        if(layer_number == 2):
            plt.title("Activation surface for node " +  str(m) +  " in output layer")
        else:
            plt.title("Activation surface for node " +  str(m) +  " in layer " +  str(layer_number + 1))
        plt.show()
```

Training model for 1 epoch

In [ ]:

```
model = Sequential()
model.add(Dense(3, input_dim=2, activation='relu'))
model.add(Dense(3, input_dim=3, activation='relu'))
model.add(Dense(3, activation='sigmoid'))
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
model.fit(X, dummy_y, epochs=1, batch_size=1)
```

```
599/599 [==============================] - 1s 1ms/step - loss: 1.1012 - accuracy: 0.3163
```

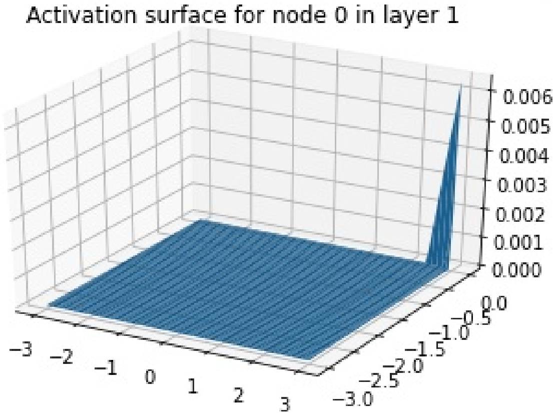
Out[ ]:

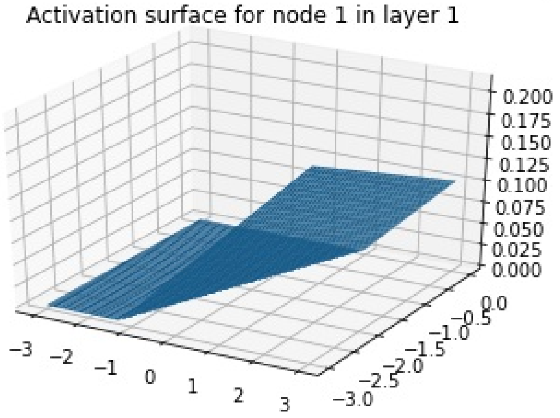
```
<tensorflow.python.keras.callbacks.History at 0x7f42e25de3d0>
```

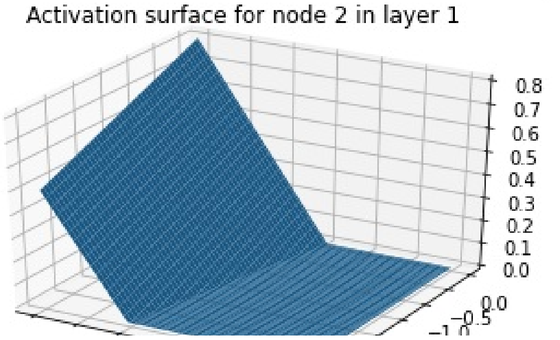
Hidden Layer 1

In [ ]:

```
plot_activation(model, 0)
```



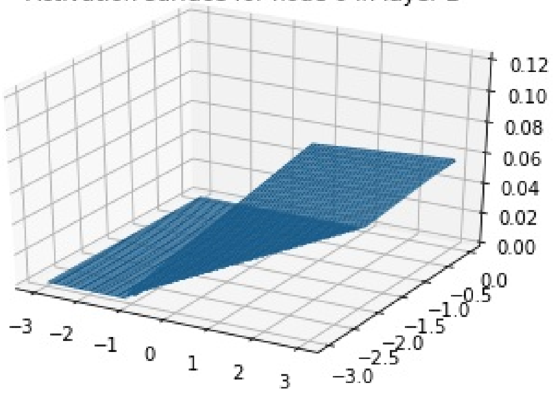


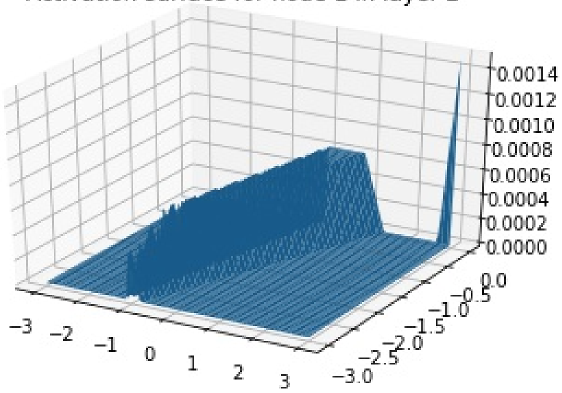


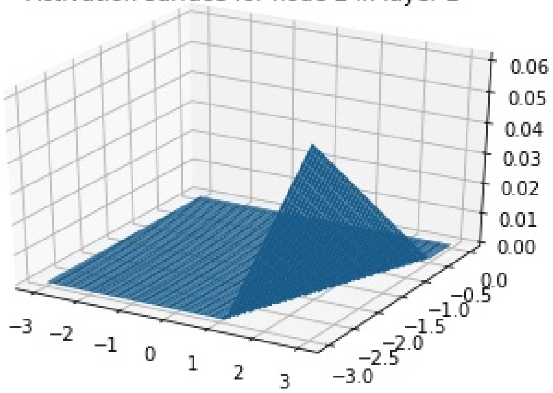
Hidden Layer 2

In [ ]:

```
plot_activation(model, 1)
```


Activation surface for node 0 in layer 2

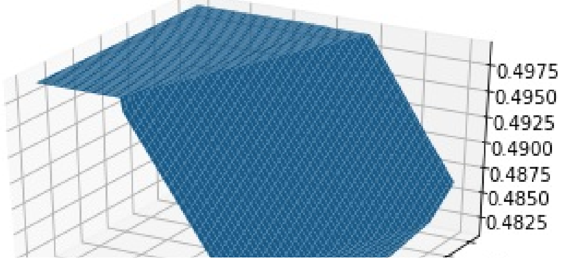

Activation surface for node 1 in layer 2

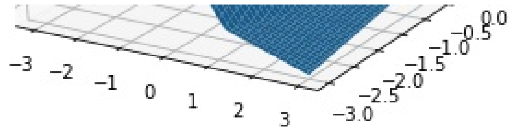

Activation surface for node 2 in layer 2

Output Layer

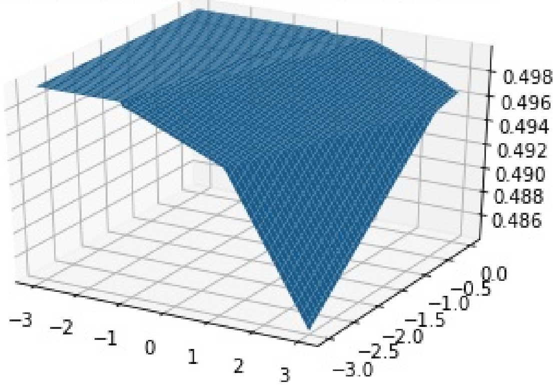
In [ ]:

```
plot_activation(model, 2)
```

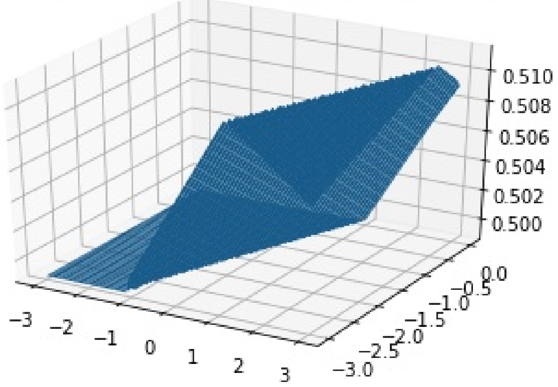

Activation surface for node 0 in output layer



Activation surface for node 1 in output layer



Activation surface for node 2 in output layer



Training model for 5 epoch

In [ ]:

```
model = Sequential()
model.add(Dense(3, input_dim=2, activation='relu'))
model.add(Dense(3, input_dim=3, activation='relu'))
model.add(Dense(3, activation='sigmoid'))
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
model.fit(X, dummy_y, epochs=5, batch_size=1)
```

```
Epoch 1/5
599/599 [==============================] - 1s 1ms/step - loss: 1.0933 - accuracy: 0.3495
Epoch 2/5
599/599 [==============================] - 1s 1ms/step - loss: 1.0827 - accuracy: 0.3406
Epoch 3/5
599/599 [==============================] - 1s 1ms/step - loss: 1.0642 - accuracy: 0.3686
Epoch 4/5
599/599 [==============================] - 1s 1ms/step - loss: 1.0431 - accuracy: 0.4395
Epoch 5/5
599/599 [==============================] - 1s 1ms/step - loss: 0.9804 - accuracy: 0.5360
```

Out[ ]:

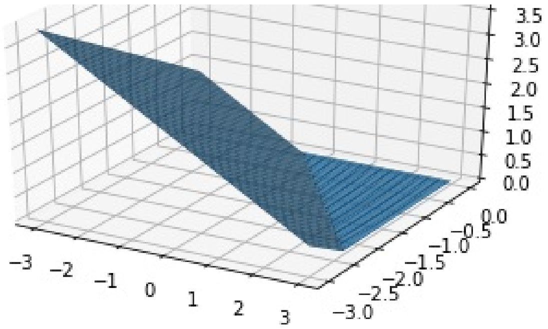
```
<tensorflow.python.keras.callbacks.History at 0x7f42e155e350>
```

Hidden Layer 1

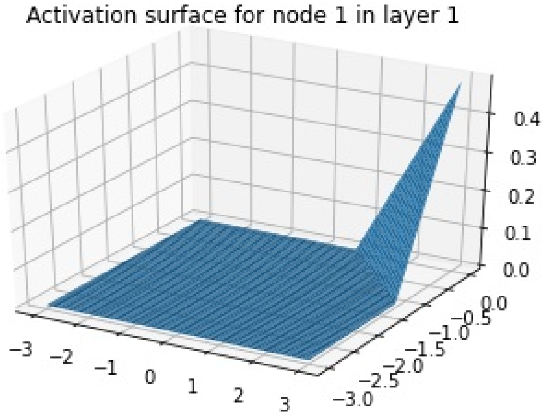
In [ ]:

```
plot_activation(model, 0)
```

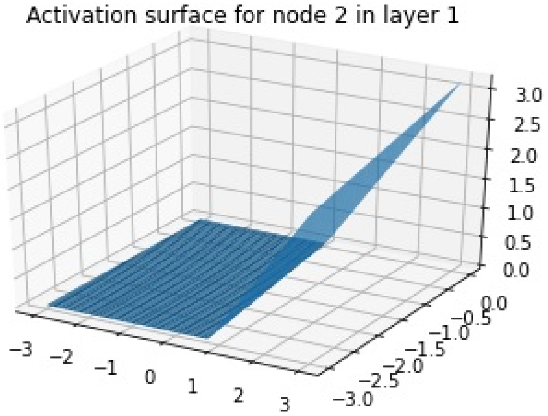
Activation surface for node 0 in layer 1



Activation surface for node 1 in layer 1



Activation surface for node 2 in layer 1

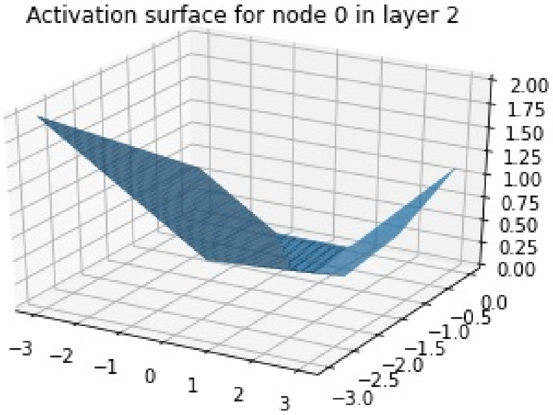


Hidden Layer 2

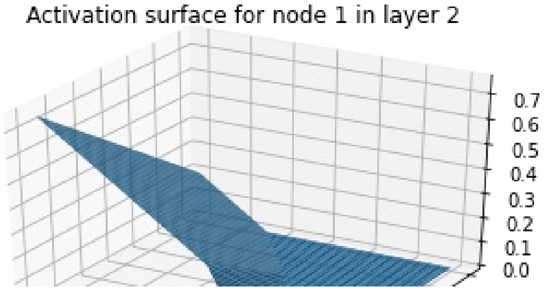
In [ ]:

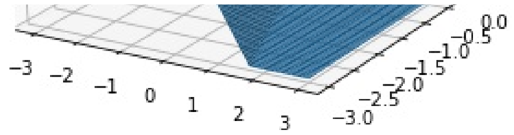
```
plot_activation(model, 1)
```

Activation surface for node 0 in layer 2

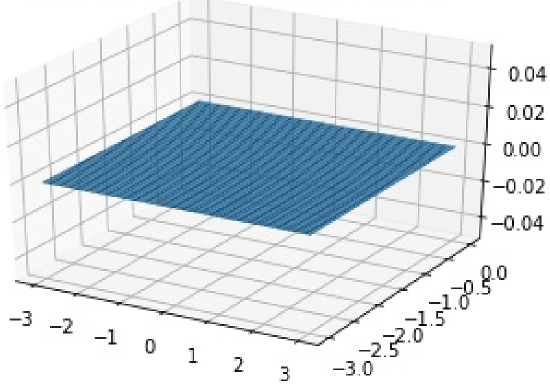


Activation surface for node 1 in layer 2





Activation surface for node 2 in layer 2

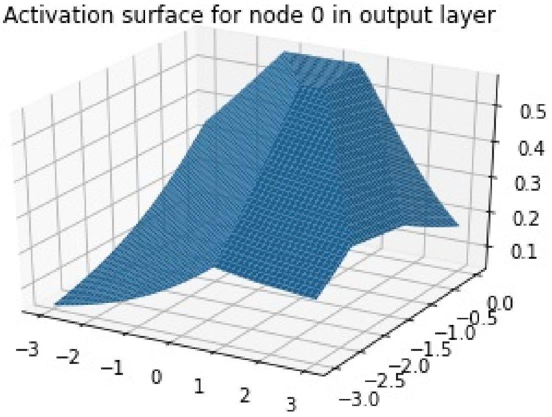


Output Layer

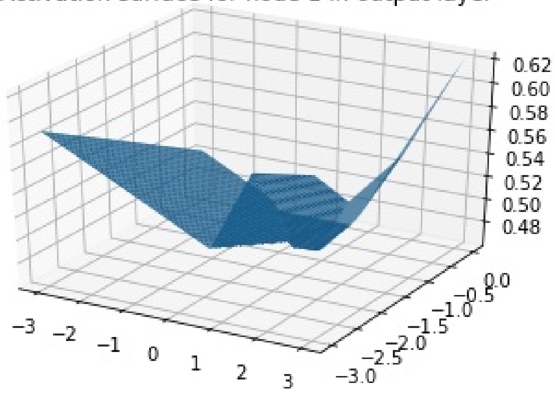
In [ ]:

```
plot_activation(model, 2)
```

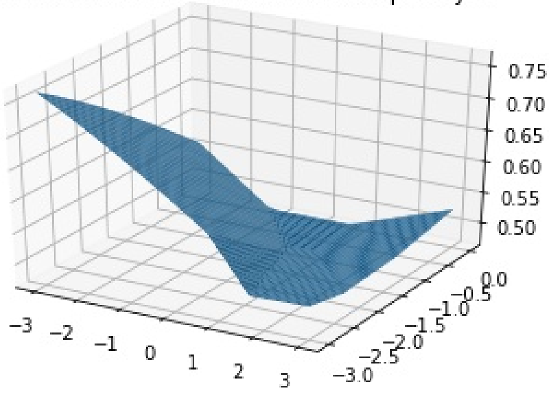
Activation surface for node 0 in output layer



Activation surface for node 1 in output layer



Activation surface for node 2 in output layer



Training model for 10 epoch

In [ ]:

```
model = Sequential()
model.add(Dense(3, input_dim=2, activation='relu'))
model.add(Dense(3, input_dim=3, activation='relu'))
model.add(Dense(3, activation='sigmoid'))
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
model.fit(X, dummy_y, epochs=10, batch_size=1)
```

```
Epoch 1/10
599/599 [==============================] - 1s 1ms/step - loss: 1.0996 - accuracy: 0.3533
Epoch 2/10
599/599 [==============================] - 1s 1ms/step - loss: 0.9939 - accuracy: 0.3993
Epoch 3/10
599/599 [==============================] - 1s 1ms/step - loss: 0.8451 - accuracy: 0.6192
Epoch 4/10
599/599 [==============================] - 1s 1ms/step - loss: 0.7493 - accuracy: 0.7002
Epoch 5/10
599/599 [==============================] - 1s 1ms/step - loss: 0.6858 - accuracy: 0.7502
Epoch 6/10
599/599 [==============================] - 1s 1ms/step - loss: 0.6117 - accuracy: 0.7577
Epoch 7/10
599/599 [==============================] - 1s 1ms/step - loss: 0.5428 - accuracy: 0.7774
Epoch 8/10
599/599 [==============================] - 1s 1ms/step - loss: 0.5241 - accuracy: 0.7876
Epoch 9/10
599/599 [==============================] - 1s 1ms/step - loss: 0.4351 - accuracy: 0.8260
Epoch 10/10
599/599 [==============================] - 1s 1ms/step - loss: 0.4397 - accuracy: 0.8141
```

Out[ ]:

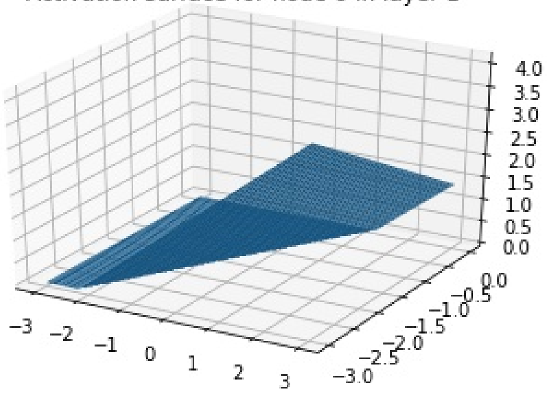
```
<tensorflow.python.keras.callbacks.History at 0x7f42e1675090>
```

Hidden Layer 1

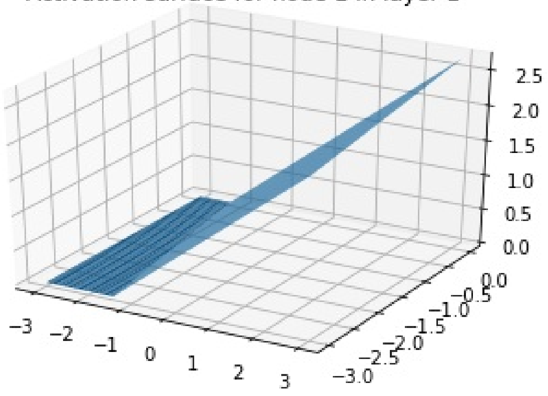
In [ ]:

```
plot_activation(model, 0)
```

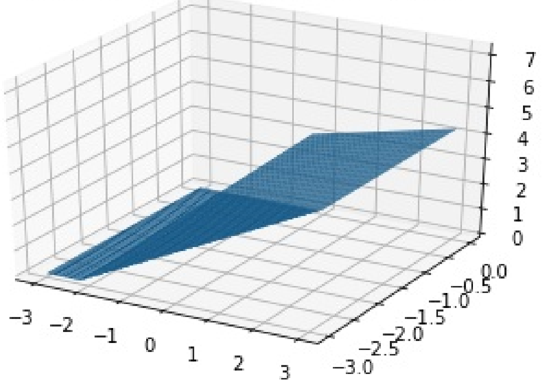
Activation surface for node 0 in layer 1



Activation surface for node 1 in layer 1



Activation surface for node 2 in layer 1

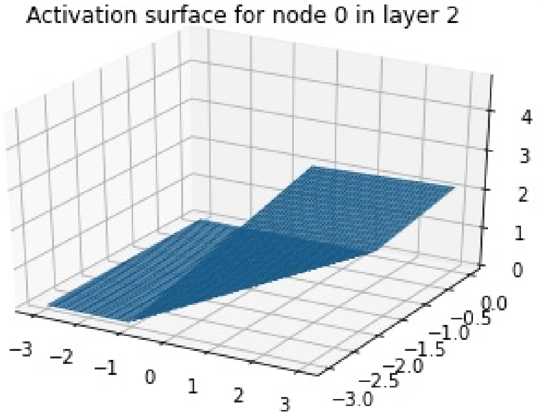


Hidden Layer 2

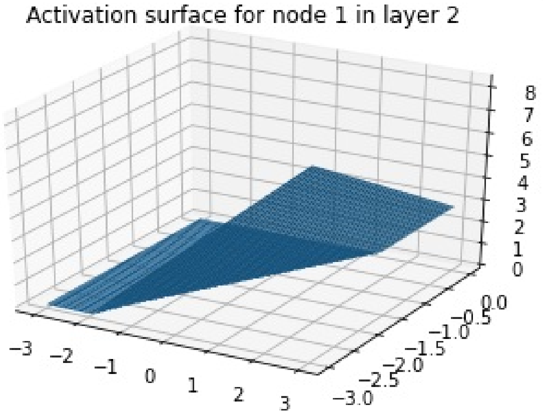
In [ ]:

```
plot_activation(model, 1)
```

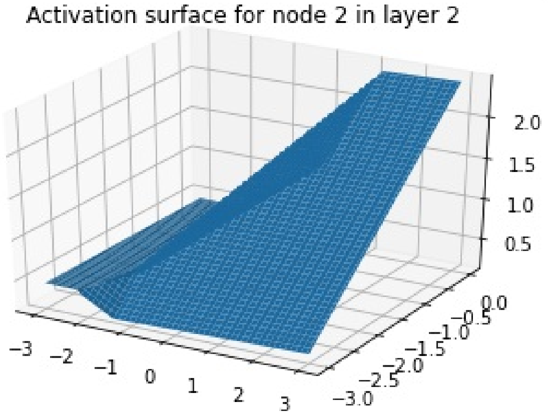
Activation surface for node 0 in layer 2



Activation surface for node 1 in layer 2



Activation surface for node 2 in layer 2

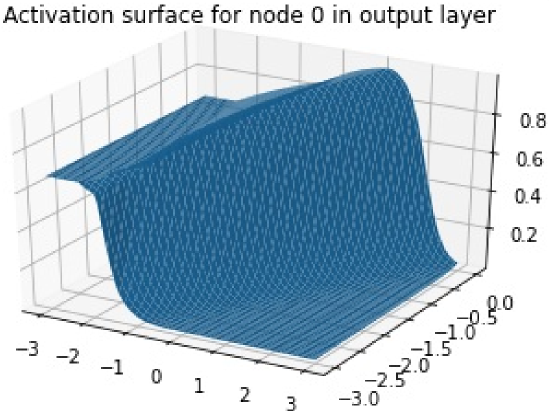


Output Layer

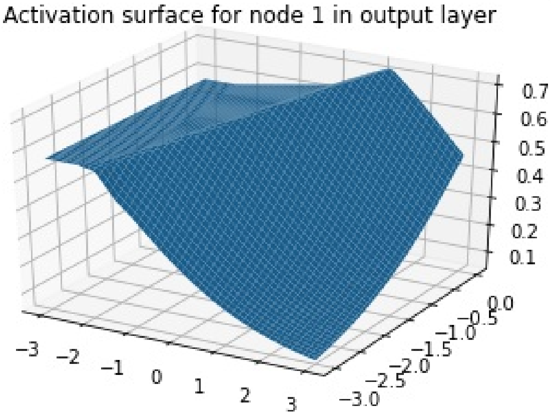
In [ ]:

```
plot_activation(model, 2)
```

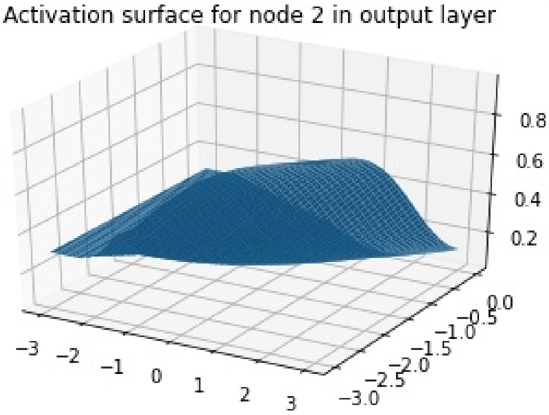
Activation surface for node 0 in output layer



Activation surface for node 1 in output layer



Activation surface for node 2 in output layer



Training model for 20 epoch

In [ ]:

```
model = Sequential()
model.add(Dense(3, input_dim=2, activation='relu'))
model.add(Dense(3, input_dim=3, activation='relu'))
model.add(Dense(3, activation='sigmoid'))
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
model.fit(X, dummy_y, epochs=20, batch_size=1)
```

```
Epoch 1/20
599/599 [==============================] - 1s 1ms/step - loss: 1.0134 - accuracy: 0.3705
Epoch 2/20
599/599 [==============================] - 1s 1ms/step - loss: 0.9400 - accuracy: 0.4548
Epoch 3/20
599/599 [==============================] - 1s 1ms/step - loss: 0.8281 - accuracy: 0.6886
Epoch 4/20
599/599 [==============================] - 1s 1ms/step - loss: 0.7511 - accuracy: 0.6951
Epoch 5/20
599/599 [==============================] - 1s 1ms/step - loss: 0.7008 - accuracy: 0.7486
Epoch 6/20
599/599 [==============================] - 1s 1ms/step - loss: 0.6591 - accuracy: 0.7386
Epoch 7/20
599/599 [==============================] - 1s 1ms/step - loss: 0.6083 - accuracy: 0.7892
```

```
Epoch 8/20
599/599 [==============================] - 1s 1ms/step - loss: 0.5327 - accuracy: 0.8332
Epoch 9/20
599/599 [==============================] - 1s 1ms/step - loss: 0.4727 - accuracy: 0.8729
Epoch 10/20
599/599 [==============================] - 1s 1ms/step - loss: 0.4282 - accuracy: 0.9303
Epoch 11/20
599/599 [==============================] - 1s 1ms/step - loss: 0.3711 - accuracy: 0.9382
Epoch 12/20
599/599 [==============================] - 1s 1ms/step - loss: 0.3302 - accuracy: 0.9484
Epoch 13/20
599/599 [==============================] - 1s 1ms/step - loss: 0.2866 - accuracy: 0.9830
Epoch 14/20
599/599 [==============================] - 1s 1ms/step - loss: 0.2533 - accuracy: 0.9930
Epoch 15/20
599/599 [==============================] - 1s 1ms/step - loss: 0.2124 - accuracy: 0.9985
Epoch 16/20
599/599 [==============================] - 1s 1ms/step - loss: 0.2012 - accuracy: 1.0000
Epoch 17/20
599/599 [==============================] - 1s 1ms/step - loss: 0.1631 - accuracy: 1.0000
Epoch 18/20
599/599 [==============================] - 1s 1ms/step - loss: 0.1404 - accuracy: 1.0000
Epoch 19/20
599/599 [==============================] - 1s 1ms/step - loss: 0.1151 - accuracy: 1.0000
Epoch 20/20
599/599 [==============================] - 1s 1ms/step - loss: 0.1014 - accuracy: 1.0000
```

Out[ ]:

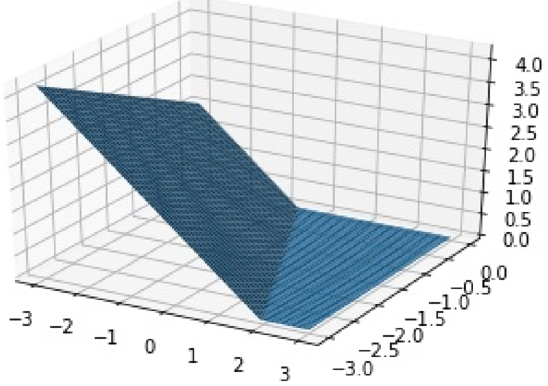
```
<tensorflow.python.keras.callbacks.History at 0x7f42de61e2d0>
```

Hidden Layer 1

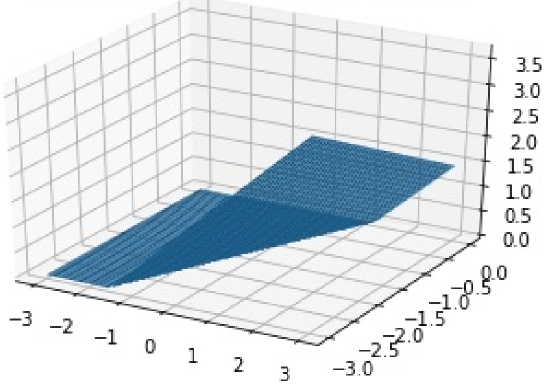
In [ ]:

```
plot_activation(model, 0)
```

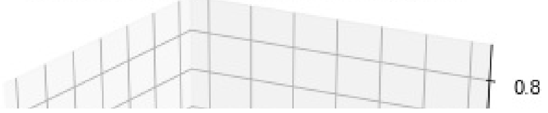
Activation surface for node 0 in layer 1

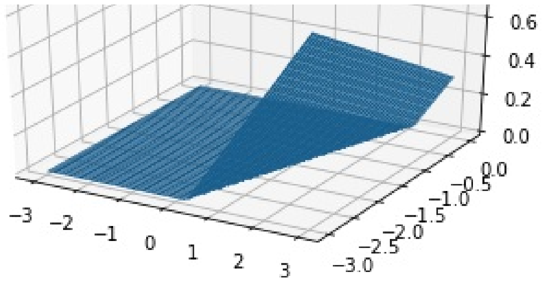


Activation surface for node 1 in layer 1



Activation surface for node 2 in layer 1



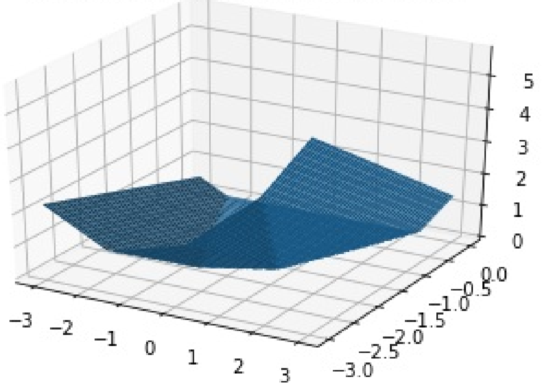


Hidden Layer 2

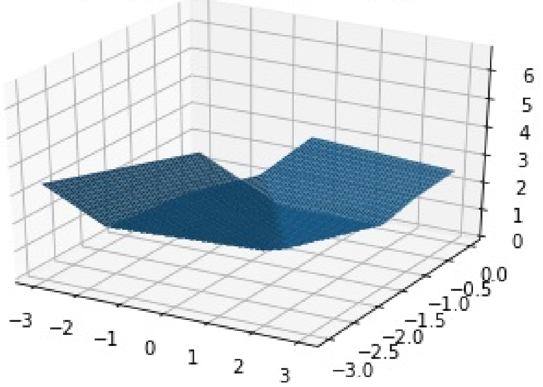
In [ ]:

```
plot_activation(model, 1)
```

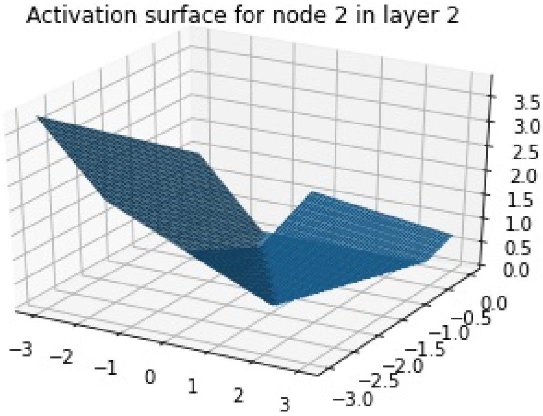
Activation surface for node 0 in layer 2



Activation surface for node 1 in layer 2



Activation surface for node 2 in layer 2

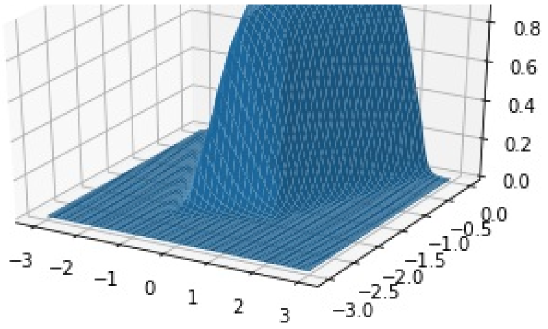


Output Layer

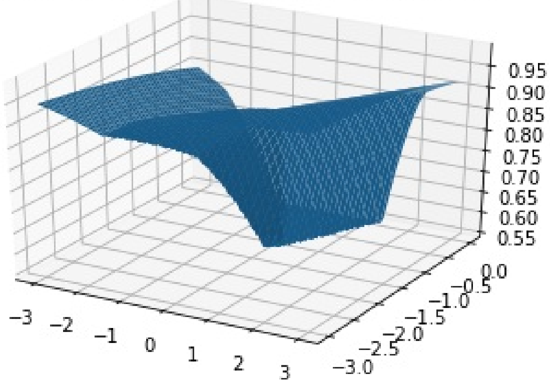
In [ ]:

```
plot_activation(model, 2)
```

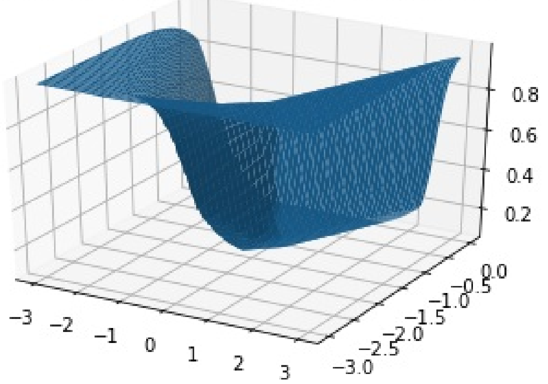
Activation surface for node 0 in output layer



Activation surface for node 1 in output layer



Activation surface for node 2 in output layer



Training model for 100 epoch

In [ ]:

```
model = Sequential()
model.add(Dense(3, input_dim=2, activation='relu'))
model.add(Dense(3, input_dim=3, activation='relu'))
model.add(Dense(3, activation='sigmoid'))
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
model.fit(X, dummy_y, epochs=100, batch_size=1)
```

```
Epoch 1/100
599/599 [==============================] - 1s 1ms/step - loss: 1.1094 - accuracy: 0.2684
Epoch 2/100
599/599 [==============================] - 1s 1ms/step - loss: 1.0992 - accuracy: 0.3217
Epoch 3/100
599/599 [==============================] - 1s 1ms/step - loss: 1.0977 - accuracy: 0.3342
Epoch 4/100
599/599 [==============================] - 1s 1ms/step - loss: 1.0526 - accuracy: 0.3167
Epoch 5/100
599/599 [==============================] - 1s 1ms/step - loss: 0.9619 - accuracy: 0.5019
Epoch 6/100
599/599 [==============================] - 1s 1ms/step - loss: 0.8772 - accuracy: 0.6076
Epoch 7/100
599/599 [==============================] - 1s 1ms/step - loss: 0.8075 - accuracy: 0.6069
Epoch 8/100
599/599 [==============================] - 1s 1ms/step - loss: 0.6976 - accuracy: 0.6900
Epoch 9/100
599/599 [==============================] - 1s 1ms/step - loss: 0.6382 - accuracy: 0.7076
Epoch 10/100
```

```
599/599 [==============================] - 1s 1ms/step - loss: 0.5829 - accuracy: 0.7960
Epoch 11/100
599/599 [==============================] - 1s 1ms/step - loss: 0.5068 - accuracy: 0.8617
Epoch 12/100
599/599 [==============================] - 1s 1ms/step - loss: 0.4690 - accuracy: 0.8943
Epoch 13/100
599/599 [==============================] - 1s 1ms/step - loss: 0.4309 - accuracy: 0.9341
Epoch 14/100
599/599 [==============================] - 1s 1ms/step - loss: 0.3902 - accuracy: 0.9543
Epoch 15/100
599/599 [==============================] - 1s 1ms/step - loss: 0.3644 - accuracy: 0.9641
Epoch 16/100
599/599 [==============================] - 1s 1ms/step - loss: 0.3091 - accuracy: 0.9690
Epoch 17/100
599/599 [==============================] - 1s 1ms/step - loss: 0.2932 - accuracy: 0.9787
Epoch 18/100
599/599 [==============================] - 1s 1ms/step - loss: 0.2651 - accuracy: 0.9880
Epoch 19/100
599/599 [==============================] - 1s 1ms/step - loss: 0.2382 - accuracy: 0.9923
Epoch 20/100
599/599 [==============================] - 1s 1ms/step - loss: 0.2184 - accuracy: 0.9964
Epoch 21/100
599/599 [==============================] - 1s 1ms/step - loss: 0.2058 - accuracy: 0.9937
Epoch 22/100
599/599 [==============================] - 1s 1ms/step - loss: 0.1816 - accuracy: 0.9926
Epoch 23/100
599/599 [==============================] - 1s 1ms/step - loss: 0.1725 - accuracy: 0.9966
Epoch 24/100
599/599 [==============================] - 1s 1ms/step - loss: 0.1474 - accuracy: 0.9938
Epoch 25/100
599/599 [==============================] - 1s 1ms/step - loss: 0.1335 - accuracy: 0.9934
Epoch 26/100
599/599 [==============================] - 1s 1ms/step - loss: 0.1250 - accuracy: 0.9909
Epoch 27/100
599/599 [==============================] - 1s 1ms/step - loss: 0.1180 - accuracy: 0.9791
Epoch 28/100
599/599 [==============================] - 1s 1ms/step - loss: 0.1119 - accuracy: 0.9887
Epoch 29/100
599/599 [==============================] - 1s 1ms/step - loss: 0.0869 - accuracy: 0.9982
Epoch 30/100
599/599 [==============================] - 1s 1ms/step - loss: 0.0738 - accuracy: 1.0000
Epoch 31/100
599/599 [==============================] - 1s 1ms/step - loss: 0.0616 - accuracy: 1.0000
Epoch 32/100
599/599 [==============================] - 1s 1ms/step - loss: 0.0570 - accuracy: 1.0000
Epoch 33/100
599/599 [==============================] - 1s 1ms/step - loss: 0.0470 - accuracy: 1.0000
Epoch 34/100
599/599 [==============================] - 1s 1ms/step - loss: 0.0428 - accuracy: 1.0000
Epoch 35/100
599/599 [==============================] - 1s 1ms/step - loss: 0.0361 - accuracy: 1.0000
Epoch 36/100
599/599 [==============================] - 1s 1ms/step - loss: 0.0344 - accuracy: 1.0000
Epoch 37/100
599/599 [==============================] - 1s 1ms/step - loss: 0.0279 - accuracy: 1.0000
Epoch 38/100
599/599 [==============================] - 1s 1ms/step - loss: 0.0260 - accuracy: 1.0000
Epoch 39/100
599/599 [==============================] - 1s 1ms/step - loss: 0.0209 - accuracy: 1.0000
Epoch 40/100
599/599 [==============================] - 1s 1ms/step - loss: 0.0173 - accuracy: 1.0000
Epoch 41/100
599/599 [==============================] - 1s 1ms/step - loss: 0.0171 - accuracy: 1.0000
Epoch 42/100
599/599 [==============================] - 1s 1ms/step - loss: 0.0137 - accuracy: 1.0000
Epoch 43/100
599/599 [==============================] - 1s 1ms/step - loss: 0.0119 - accuracy: 1.0000
Epoch 44/100
599/599 [==============================] - 1s 1ms/step - loss: 0.0102 - accuracy: 1.0000
Epoch 45/100
599/599 [==============================] - 1s 1ms/step - loss: 0.0101 - accuracy: 1.0000
Epoch 46/100
```

```
599/599 [==============================] - 1s 1ms/step - loss: 0.0101 - accuracy: 1.0000
Epoch 47/100
599/599 [==============================] - 1s 1ms/step - loss: 0.0077 - accuracy: 1.0000
Epoch 48/100
599/599 [==============================] - 1s 1ms/step - loss: 0.0064 - accuracy: 1.0000
Epoch 49/100
599/599 [==============================] - 1s 1ms/step - loss: 0.0052 - accuracy: 1.0000
Epoch 50/100
599/599 [==============================] - 1s 1ms/step - loss: 0.0055 - accuracy: 1.0000
Epoch 51/100
599/599 [==============================] - 1s 1ms/step - loss: 0.0051 - accuracy: 1.0000
Epoch 52/100
599/599 [==============================] - 1s 1ms/step - loss: 0.0043 - accuracy: 1.0000
Epoch 53/100
599/599 [==============================] - 1s 1ms/step - loss: 0.0043 - accuracy: 1.0000
Epoch 54/100
599/599 [==============================] - 1s 1ms/step - loss: 0.0034 - accuracy: 1.0000
Epoch 55/100
599/599 [==============================] - 1s 1ms/step - loss: 0.0030 - accuracy: 1.0000
Epoch 56/100
599/599 [==============================] - 1s 1ms/step - loss: 0.0024 - accuracy: 1.0000
Epoch 57/100
599/599 [==============================] - 1s 1ms/step - loss: 0.0023 - accuracy: 1.0000
Epoch 58/100
599/599 [==============================] - 1s 1ms/step - loss: 0.0023 - accuracy: 1.0000
Epoch 59/100
599/599 [==============================] - 1s 1ms/step - loss: 0.0022 - accuracy: 1.0000
Epoch 60/100
599/599 [==============================] - 1s 1ms/step - loss: 0.0017 - accuracy: 1.0000
Epoch 61/100
599/599 [==============================] - 1s 1ms/step - loss: 0.0018 - accuracy: 1.0000
Epoch 62/100
599/599 [==============================] - 1s 1ms/step - loss: 0.0013 - accuracy: 1.0000
Epoch 63/100
599/599 [==============================] - 1s 1ms/step - loss: 0.0013 - accuracy: 1.0000
Epoch 64/100
599/599 [==============================] - 1s 1ms/step - loss: 0.0012 - accuracy: 1.0000
Epoch 65/100
599/599 [==============================] - 1s 1ms/step - loss: 0.0013 - accuracy: 1.0000
Epoch 66/100
599/599 [==============================] - 1s 1ms/step - loss: 9.9600e-04 - accuracy: 1.0
000
Epoch 67/100
599/599 [==============================] - 1s 1ms/step - loss: 0.0011 - accuracy: 1.0000
Epoch 68/100
599/599 [==============================] - 1s 1ms/step - loss: 0.0011 - accuracy: 1.0000
Epoch 69/100
599/599 [==============================] - 1s 1ms/step - loss: 7.0234e-04 - accuracy: 1.0
000
Epoch 70/100
599/599 [==============================] - 1s 1ms/step - loss: 8.2760e-04 - accuracy: 1.0
000
Epoch 71/100
599/599 [==============================] - 1s 1ms/step - loss: 7.2125e-04 - accuracy: 1.0
000
Epoch 72/100
599/599 [==============================] - 1s 1ms/step - loss: 7.8294e-04 - accuracy: 1.0
000
Epoch 73/100
599/599 [==============================] - 1s 1ms/step - loss: 5.1601e-04 - accuracy: 1.0
000
Epoch 74/100
599/599 [==============================] - 1s 1ms/step - loss: 4.7975e-04 - accuracy: 1.0
000
Epoch 75/100
599/599 [==============================] - 1s 1ms/step - loss: 5.2099e-04 - accuracy: 1.0
000
Epoch 76/100
599/599 [==============================] - 1s 1ms/step - loss: 0.0013 - accuracy: 1.0000
Epoch 77/100
599/599 [==============================] - 1s 1ms/step - loss: 4.3232e-04 - accuracy: 1.0
000
```

```
Epoch 78/100
599/599 [==============================] - 1s 1ms/step - loss: 3.7577e-04 - accuracy: 1.0
000
Epoch 79/100
599/599 [==============================] - 1s 1ms/step - loss: 5.4939e-04 - accuracy: 1.0
000
Epoch 80/100
599/599 [==============================] - 1s 1ms/step - loss: 5.0633e-04 - accuracy: 1.0
000
Epoch 81/100
599/599 [==============================] - 1s 1ms/step - loss: 3.4182e-04 - accuracy: 1.0
000
Epoch 82/100
599/599 [==============================] - 1s 1ms/step - loss: 5.1992e-04 - accuracy: 1.0
000
Epoch 83/100
599/599 [==============================] - 1s 1ms/step - loss: 3.0520e-04 - accuracy: 1.0
000
Epoch 84/100
599/599 [==============================] - 1s 1ms/step - loss: 3.3217e-04 - accuracy: 1.0
000
Epoch 85/100
599/599 [==============================] - 1s 1ms/step - loss: 3.3882e-04 - accuracy: 1.0
000
Epoch 86/100
599/599 [==============================] - 1s 1ms/step - loss: 2.4614e-04 - accuracy: 1.0
000
Epoch 87/100
599/599 [==============================] - 1s 1ms/step - loss: 2.8898e-04 - accuracy: 1.0
000
Epoch 88/100
599/599 [==============================] - 1s 1ms/step - loss: 2.3821e-04 - accuracy: 1.0
000
Epoch 89/100
599/599 [==============================] - 1s 1ms/step - loss: 1.7585e-04 - accuracy: 1.0
000
Epoch 90/100
599/599 [==============================] - 1s 1ms/step - loss: 3.5562e-04 - accuracy: 1.0
000
Epoch 91/100
599/599 [==============================] - 1s 1ms/step - loss: 1.8536e-04 - accuracy: 1.0
000
Epoch 92/100
599/599 [==============================] - 1s 1ms/step - loss: 2.4623e-04 - accuracy: 1.0
000
Epoch 93/100
599/599 [==============================] - 1s 1ms/step - loss: 2.1548e-04 - accuracy: 1.0
000
Epoch 94/100
599/599 [==============================] - 1s 1ms/step - loss: 1.6628e-04 - accuracy: 1.0
000
Epoch 95/100
599/599 [==============================] - 1s 1ms/step - loss: 1.2861e-04 - accuracy: 1.0
000
Epoch 96/100
599/599 [==============================] - 1s 1ms/step - loss: 1.5759e-04 - accuracy: 1.0
000
Epoch 97/100
599/599 [==============================] - 1s 1ms/step - loss: 1.2613e-04 - accuracy: 1.0
000
Epoch 98/100
599/599 [==============================] - 1s 1ms/step - loss: 1.8827e-04 - accuracy: 1.0
000
Epoch 99/100
599/599 [==============================] - 1s 1ms/step - loss: 1.9659e-04 - accuracy: 1.0
000
Epoch 100/100
599/599 [==============================] - 1s 1ms/step - loss: 1.2131e-04 - accuracy: 1.0
000
```

```
Out[ ]:
```

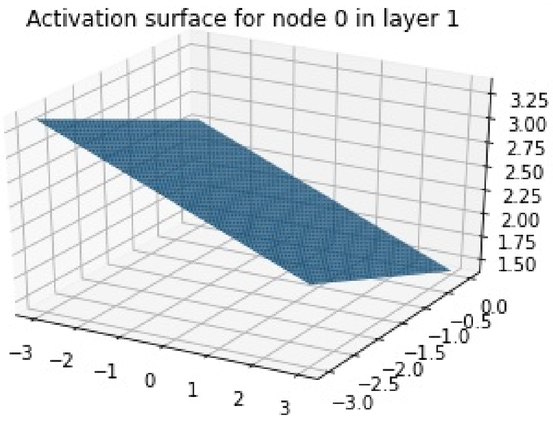
```
<tensorflow.python.keras.callbacks.History at 0x7f42de5aba10>
```

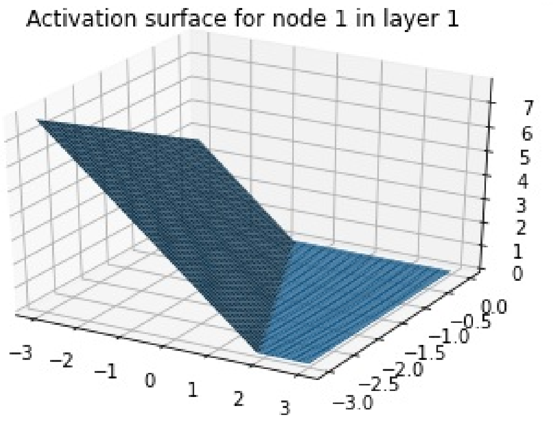
```
<tensorflow.python.keras.callbacks.History at 0x7f42de5aba10>
```

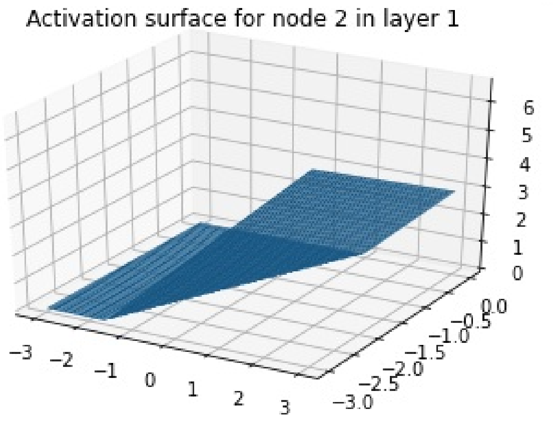
Hidden Layer 1

In [ ]:

```
plot_activation(model, 0)
```



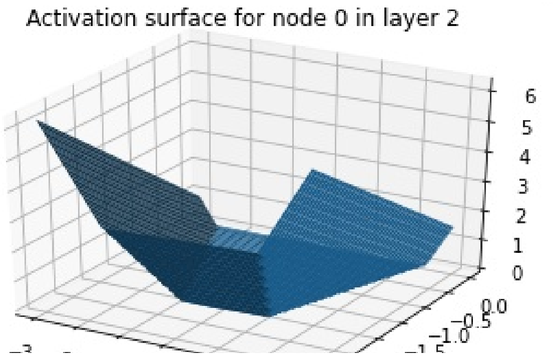


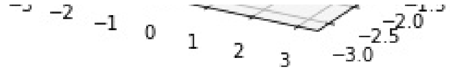


Hidden Layer 2

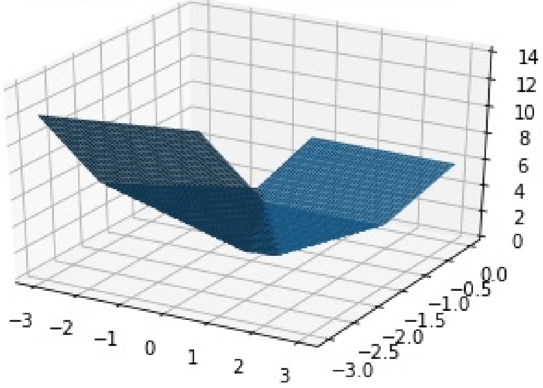
In [ ]:

```
plot_activation(model, 1)
```

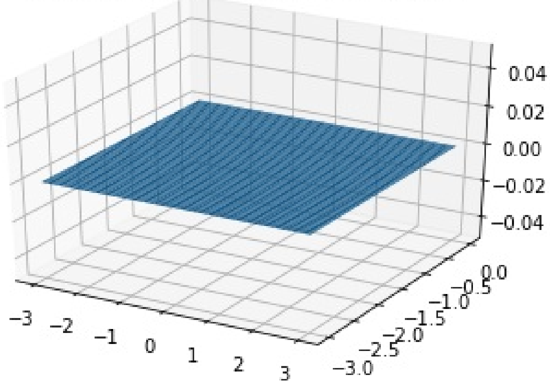




Activation surface for node 1 in layer 2



Activation surface for node 2 in layer 2

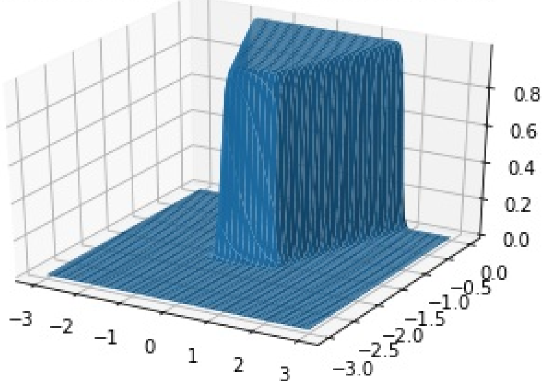


Output Layer

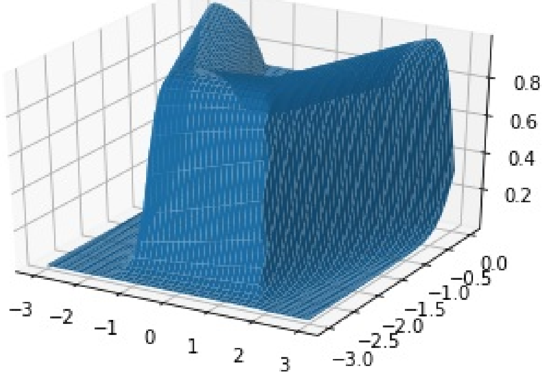
In [ ]:

```
plot_activation(model, 2)
```

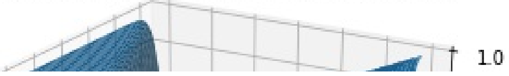
Activation surface for node 0 in output layer

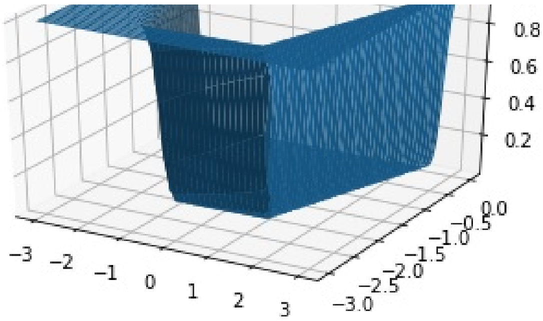


Activation surface for node 1 in output layer



Activation surface for node 2 in output layer




0.8
0.6
0.4
0.2
−3
−2
−1
0
1
2
3
0.0
−0.5
−1.0
−1.5
−2.0
−2.5
−3.0