Hands-on Exercise CLASS Module

In [1]:

```
!pip install --user mlxtend
```

```
Requirement already satisfied: mlxtend in /users/PES0801/dogipakr/.local/l
ib/python3.6/site-packages
Requirement already satisfied: pandas>=0.24.2 in /users/PES0801/dogipakr/.
local/lib/python3.6/site-packages (from mlxtend)
Requirement already satisfied: numpy>=1.16.2 in /users/PES0801/dogipakr/.l
ocal/lib/python3.6/site-packages (from mlxtend)
Requirement already satisfied: scipy>=1.2.1 in /users/PES0801/dogipakr/.lo
cal/lib/python3.6/site-packages (from mlxtend)
Requirement already satisfied: joblib>=0.13.2 in /users/PES0801/dogipakr/.
local/lib/python3.6/site-packages (from mlxtend)
Requirement already satisfied: scikit-learn>=0.20.3 in /users/PES0801/dogi
pakr/.local/lib/python3.6/site-packages (from mlxtend)
Requirement already satisfied: matplotlib>=3.0.0 in /users/PES0801/dogipak
r/.local/lib/python3.6/site-packages (from mlxtend)
Requirement already satisfied: setuptools in /usr/local/anaconda5/lib/pyth
on3.6/site-packages (from mlxtend)
Requirement already satisfied: pytz>=2017.2 in /usr/local/anaconda5/lib/py
thon3.6/site-packages (from pandas>=0.24.2->mlxtend)
Requirement already satisfied: python-dateutil>=2.6.1 in /usr/local/anacon
da5/lib/python3.6/site-packages (from pandas>=0.24.2->mlxtend)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in
/usr/local/anaconda5/lib/python3.6/site-packages (from matplotlib>=3.0.0->
mlxtend)
Requirement already satisfied: cycler>=0.10 in /usr/local/anaconda5/lib/py
thon3.6/site-packages (from matplotlib>=3.0.0->mlxtend)
Requirement already satisfied: kiwisolver>=1.0.1 in /users/PES0801/dogipak
r/.local/lib/python3.6/site-packages (from matplotlib>=3.0.0->mlxtend)
Requirement already satisfied: six>=1.5 in /usr/local/anaconda5/lib/python
3.6/site-packages (from python-dateutil>=2.6.1->pandas>=0.24.2->mlxtend)
You are using pip version 9.0.1, however version 19.3.1 is available.
You should consider upgrading via the 'pip install --upgrade pip' command.
```

In [2]:

```
import numpy as np

#Plotting packages
import matplotlib.pyplot as plt
import seaborn as sns

#Classification Algorithms
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC

#Ensemble Methods
from sklearn.ensemble import BaggingClassifier
from sklearn.ensemble import BaggingRegressor
from sklearn.model_selection import cross_val_score, train_test_split
from sklearn.ensemble import AdaBoostClassifier

#Mlxtend for visualizing classification decision boundaries
from mlxtend.plotting import plot_decision_regions
```

In [3]:

```
# Generating Data1

np.random.seed(100)

a = np.random.multivariate_normal([2,2],[[0.5,0], [0,0.5]], 200)
b = np.random.multivariate_normal([4,4],[[0.5,0], [0,0.5]], 200)

Data1_X = np.vstack((a,b))
Data1_Y = np.hstack((np.ones(200).T,np.zeros(200).T)).astype(int)


# Generating Data2

np.random.seed(100)

a1 = np.random.multivariate_normal([2,2],[[0.25,0], [0,0.25]],200)
a2 = np.random.multivariate_normal([2,4],[[0.25,0], [0,0.25]],200)
a3 = np.random.multivariate_normal([4,2],[[0.25,0], [0,0.25]],200)
a4 = np.random.multivariate_normal([4,4],[[0.25,0], [0,0.25]],200)

Data2_X = np.vstack((a1,a4,a2,a3))
Data2_Y = np.hstack((np.ones(400).T,np.zeros(400).T)).astype(int)


# Generating Data3

np.random.seed(100)

a1 = np.random.uniform(4,6,[200,2])
a2 = np.random.uniform(0,10,[200,2])

Data3_X = np.vstack((a1,a2))
Data3_Y = np.hstack((np.ones(200).T,np.zeros(200).T)).astype(int)


# Generating Data4

np.random.seed(100)

Data4_X = np.random.uniform(0,12,[500,2])
Data4_Y = np.ones([500]).astype(int)
Data4_Y[np.multiply(Data4_X[:,0],Data4_X[:,0]) + np.multiply(Data4_X[:,1],Data4_X[:,1])
- 100 < 0 ] = 0
```

1. Decision Tree

Use **Data3** to answer the following questions.

Question 1a: Compute and print the 10-fold cross-validation accuracy using decision tree classifiers with max_depth = 2,4,6,8,10, and 50.

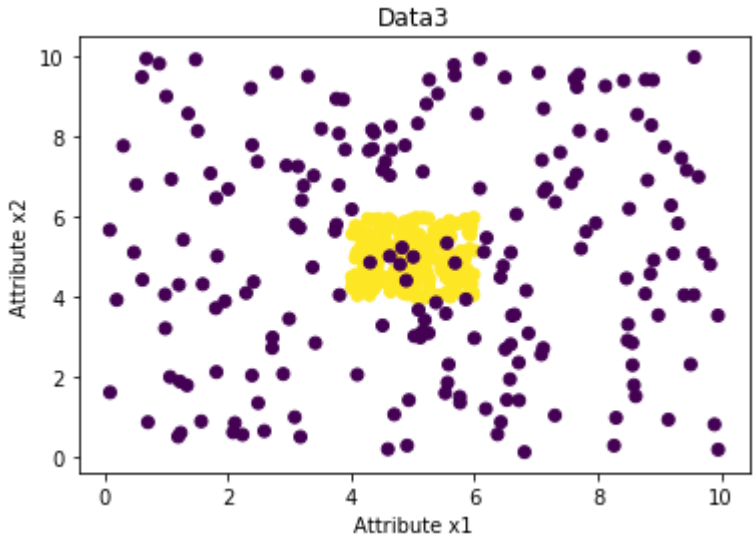
In [4]:

```
#Data3_X.shape
depth=[2,4,6,8,10,50]
for i in depth :
    dt = DecisionTreeClassifier(max_depth=i)
    dt_scores = cross_val_score(dt, Data3_X, Data3_Y, cv=10, scoring='accuracy')
    print("The accuracy of the decision tree classifier with max_depth =" +str(i)+ " wo
uld be " +str(dt_scores.mean()*100) )
```

```
The accuracy of the decision tree classifier with max_depth =2 would be 8
7.5
The accuracy of the decision tree classifier with max_depth =4 would be 9
7.0
The accuracy of the decision tree classifier with max_depth =6 would be 9
6.74999999999999
The accuracy of the decision tree classifier with max_depth =8 would be 9
4.99999999999999
The accuracy of the decision tree classifier with max_depth =10 would be 9
4.24999999999999
The accuracy of the decision tree classifier with max_depth =50 would be 9
4.5
```

In [5]:

```
plt.scatter(Data3_X[:,0],Data3_X[:,1], c=Data3_Y)
plt.xlabel('Attribute x1')
plt.ylabel('Attribute x2')
plt.title('Data3')
plt.show()
```



Question 1b: For what values of max_depth did you observe the lowest accuracy? What is this phenomenon called?

Answer: For a depth of 2, i have observed the lowest accuracy for the decision tree and this phenemenon is called underfitting.

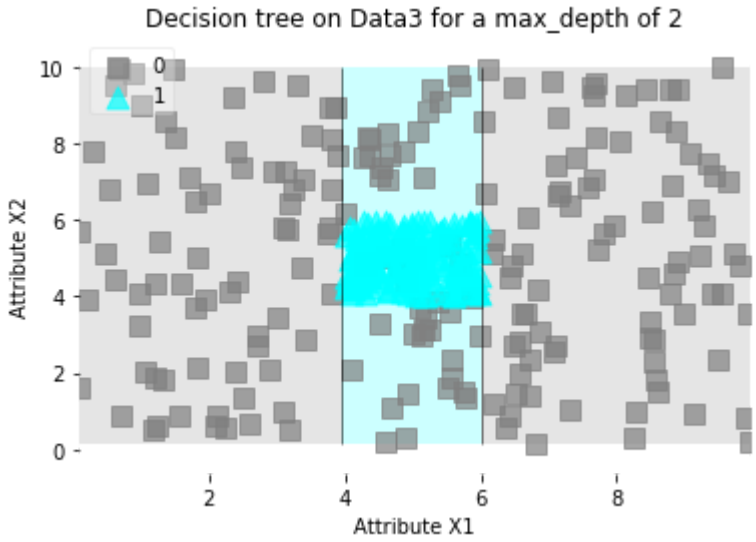
Question 1c: What accuracy did you observe for max depth=50? What is the difference between this accuracy and the highest accuracy? What is this phenomenon called?

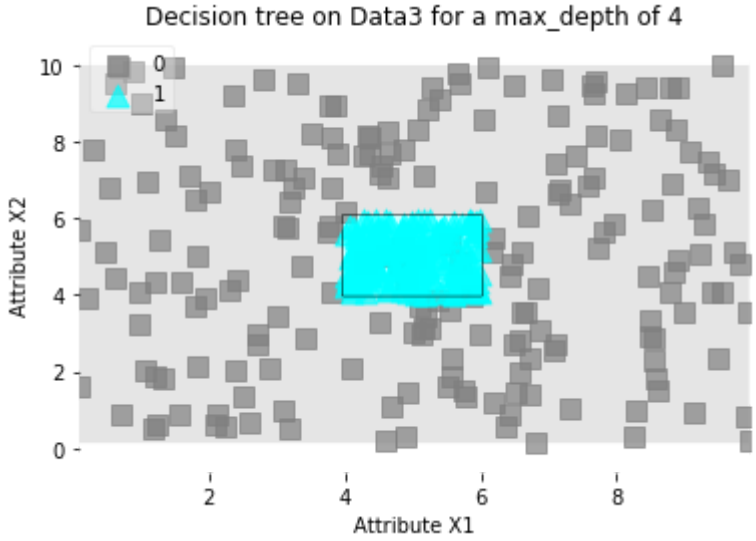
Answer: 94.5 accuracy has been obsereved for max_depth=50 and difference in accuracy between highest accuracy and the accuracy at depth=50 is around 2.5 and this phenomenon is called overfitting .

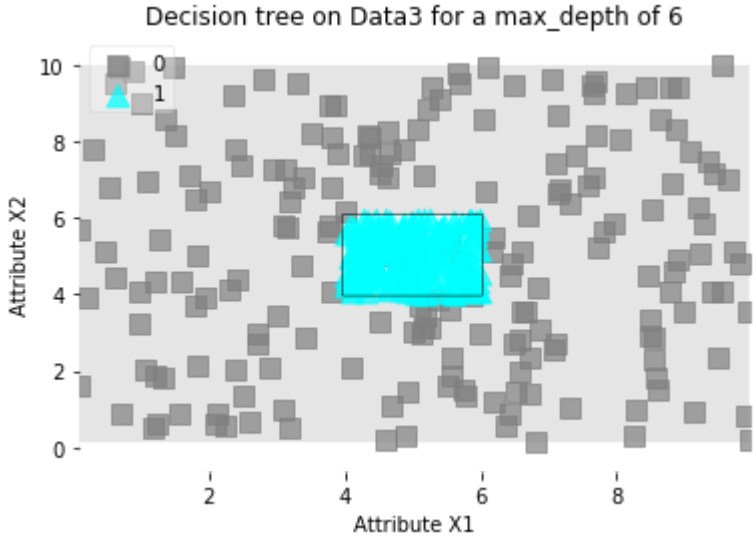
Question 1d: Plot decision regions for the above decision tree models

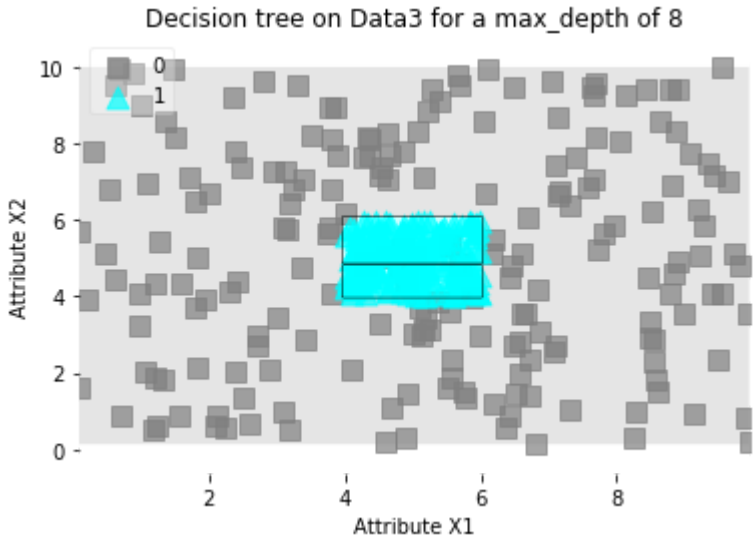
In [41]:

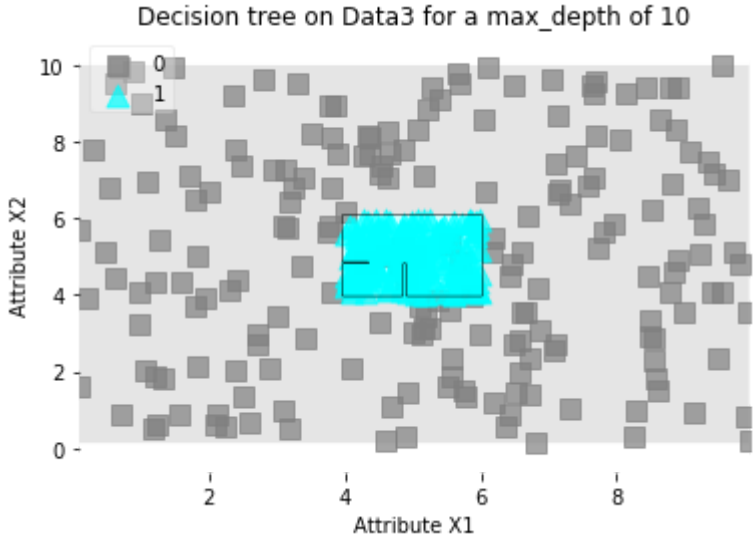
```
depth=[2,4,6,8,10,50]
scatter_kwargs = {'s': 120, 'edgecolor': None, 'alpha': 0.7}
scatter_highlight_kwargs = {'s': 120, 'label': 'Test data', 'alpha': 0.7}
for i in depth :
    dt = DecisionTreeClassifier(max_depth=i)
    dt.fit(Data3_X, Data3_Y)
    plot_decision_regions(Data3_X, Data3_Y, clf=dt,colors='gray,cyan',scatter_kwargs=sc
atter_kwargs,
                      contourf_kwargs=contourf_kwargs,
                      scatter_highlight_kwargs=scatter_highlight_kwargs,legend=2,zoom_f
actor=1000)
    plt.xlabel('Attribute X1')
    plt.ylabel('Attribute X2')
    plt.title('Decision tree on Data3 for a max_depth of '+ str(i) )
    plt.show()
```

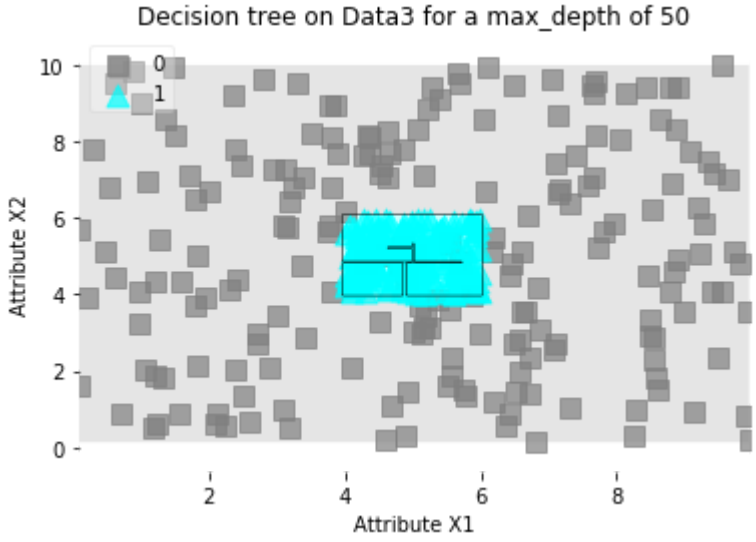

Decision tree on Data3 for a max_depth of 2
0
1
Attribute X2
Attribute X1
0
2
4
6
8
10


Decision tree on Data3 for a max_depth of 4
0
1
Attribute X2
Attribute X1
0
2
4
6
8
10


Decision tree on Data3 for a max_depth of 6
0
1
Attribute X2
Attribute X1
0
2
4
6
8
10


Decision tree on Data3 for a max_depth of 8
0
1
Attribute X2
Attribute X1
0
2
4
6
8
10


Decision tree on Data3 for a max_depth of 10
0
1
Attribute X2
Attribute X1
0
2
4
6
8
10


Decision tree on Data3 for a max_depth of 50
0
1
Attribute X2
Attribute X1
0
2
4
6
8
10

Question 1e: Based on the decision regions, which depth is better suited for this data? Explain your reason.

Answer: Depth 4 is better suited for this data as we would see the decision boundary seperated the two classes correctly ,model is at tradeoff point i.e the hypothesis is generalized and we the hyper parameter max_depth greater than 4, we could the lines parallel to the axis ,more number boundaries and those models are overfitted for this data.

2. k Nearest Neighbor

Use **Data2** to answer the following questions.

Question 2a: Compute and print the 10-fold cross-validation accuracy for a kNN classifier with n_neighbors = 1, 5, 10, 50

In [50]:

```
K=[1,5,10,50,100,120,150,200]
for i in K :
    KNN= KNeighborsClassifier(n_neighbors=i)
    KNN_scores = cross_val_score(KNN, Data2_X, Data2_Y, cv=10, scoring='accuracy')
    print("The accuracy of the KNN classifier with K value =" +str(i)+ " would be " +st
r(KNN_scores.mean()*100) )
```

```
The accuracy of the KNN classifier with K value =1 would be 91.25
The accuracy of the KNN classifier with K value =5 would be 93.5
The accuracy of the KNN classifier with K value =10 would be 94.0
The accuracy of the KNN classifier with K value =50 would be 94.1249999999
9999
The accuracy of the KNN classifier with K value =100 would be 94.249999999
99999
The accuracy of the KNN classifier with K value =120 would be 94.124999999
99999
The accuracy of the KNN classifier with K value =150 would be 93.999999999
99999
The accuracy of the KNN classifier with K value =200 would be 93.375000000
00001
```

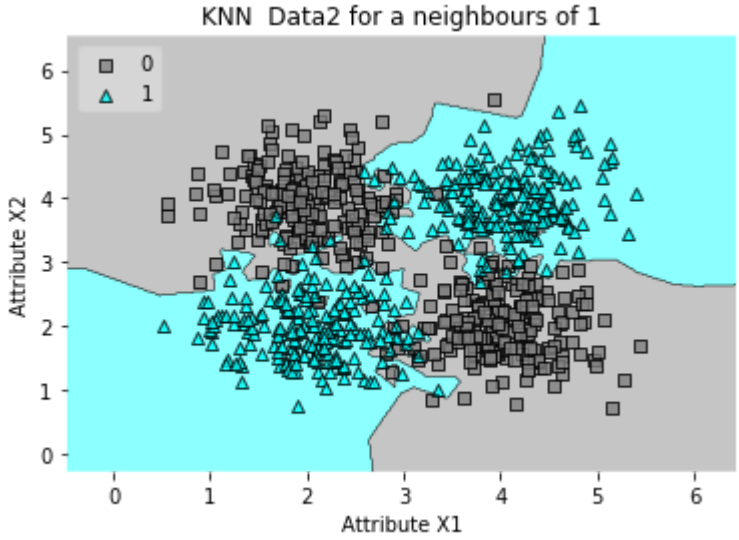
Question 2b: For what values of n_neighbors did you observe the lowest accuracy? What is this phenomenon called?

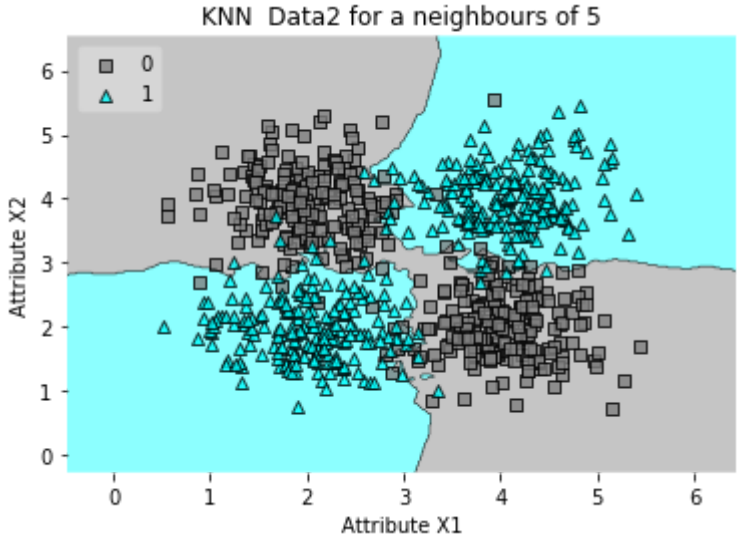
Answer: when k=1, I have obsereved the lowest accuracy and this phenomenon is due to overfitting ,when checked for the k values greater than 50 I have observed there will be potential trade off for this model when choosing the value in between 100 and 120.

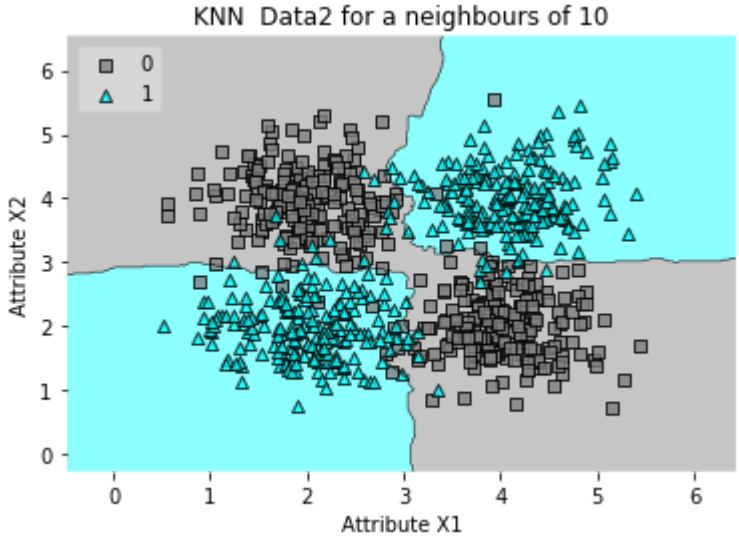
Question 2c: Plot decision regions for a kNN classifier with n_neighbors = 1, 5, 10, 50

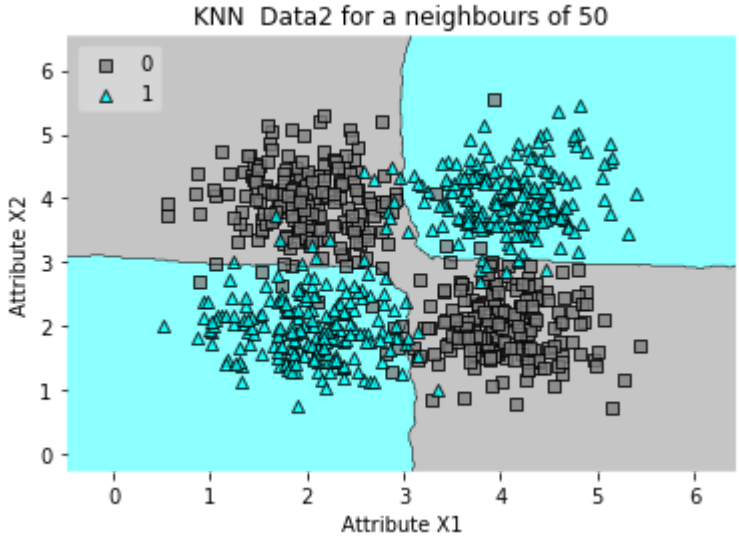
In [48]:

```
#K=[1,5,10,50,120]
K=[1,5,10,50]
for i in K :
    KNN= KNeighborsClassifier(n_neighbors=i)
    KNN.fit(Data2_X, Data2_Y)
    plot_decision_regions(Data2_X, Data2_Y, clf=KNN,colors='gray,cyan', legend=2)
    plt.xlabel('Attribute X1')
    plt.ylabel('Attribute X2')
    plt.title('KNN  Data2 for a neighbours of '+ str(i) )
    plt.show( )
```


KNN Data2 for a neighbours of 1
0
1
Attribute X2
Attribute X1
0
1
2
3
4
5
6


KNN Data2 for a neighbours of 5
0
1
Attribute X2
Attribute X1
0
1
2
3
4
5
6


KNN Data2 for a neighbours of 10
0
1
Attribute X2
Attribute X1
0
1
2
3
4
5
6



Question 2d: From the plots for **Question 2c** what do you notice about the nature of decision boundary as the n_neighbors are increasing.

Answer: when the K (neighbour) value increases the decision boundary becomes smoother. From the above graphs when k=1 the decision boundary edges were irregular where as for the k value 50 the decision boundary edges were smoother.

3. Naive Bayes

Question 3a: Compute and print the 10-fold cross-validation accuracy for a NB classifier on all four datasets: Data1, Data2, Data3, Data4

In [9]:

```
nb1 = GaussianNB()
nb_scores = cross_val_score(nb1, Data1_X, Data1_Y, cv=10, scoring='accuracy')
[nb_scores.mean(), nb_scores.std()]
```

Out[9]:

```
[0.9675, 0.03172144385112379]
```

In [10]:

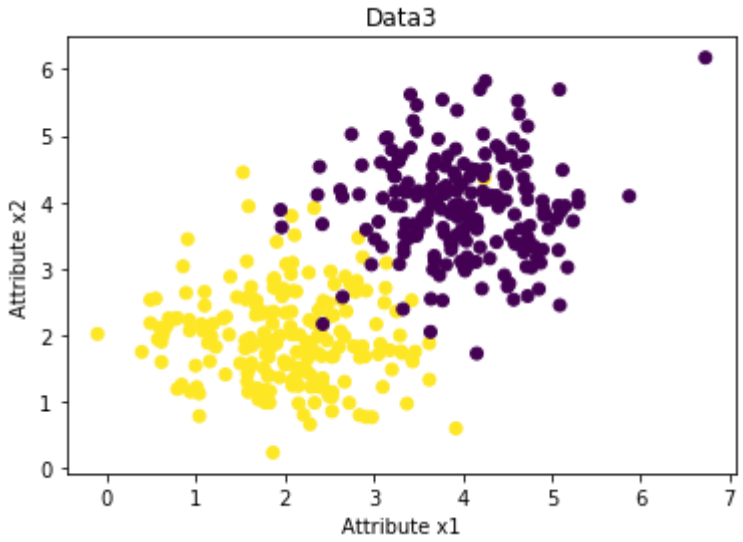
```
nb2 = GaussianNB()
nb_scores = cross_val_score(nb2, Data2_X, Data2_Y, cv=10, scoring='accuracy')
[nb_scores.mean(), nb_scores.std()]
```

Out[10]:

```
[0.049999999999999996, 0.026809513236909017]
```

In [11]:

```
plt.scatter(Data1_X[:,0],Data1_X[:,1], c=Data1_Y)
plt.xlabel('Attribute x1')
plt.ylabel('Attribute x2')
plt.title('Data3')
plt.show()
```



In [12]:

```
nb3 = GaussianNB()
nb_scores = cross_val_score(nb3, Data3_X, Data3_Y, cv=10, scoring='accuracy')
[nb_scores.mean(), nb_scores.std()]
```

Out[12]:

```
[0.96, 0.027838821814150098]
```

In [13]:

```
nb4 = GaussianNB()
nb_scores = cross_val_score(nb4, Data4_X, Data4_Y, cv=10, scoring='accuracy')
[nb_scores.mean(), nb_scores.std()]
```

Out[13]:

```
[0.9640736294517807, 0.026052087140989725]
```

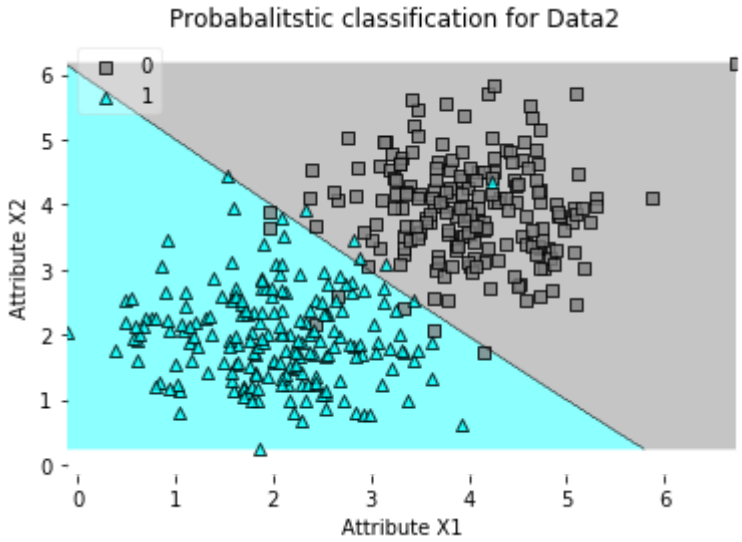
Question 3b: State your observations on the datasets the NB algorithm performed poorly.

Answer: On the data2 the accuracy of the NB classifier performed poorly.

Question 3c: Plot decision regions for a NB classifier on each of the four datasets

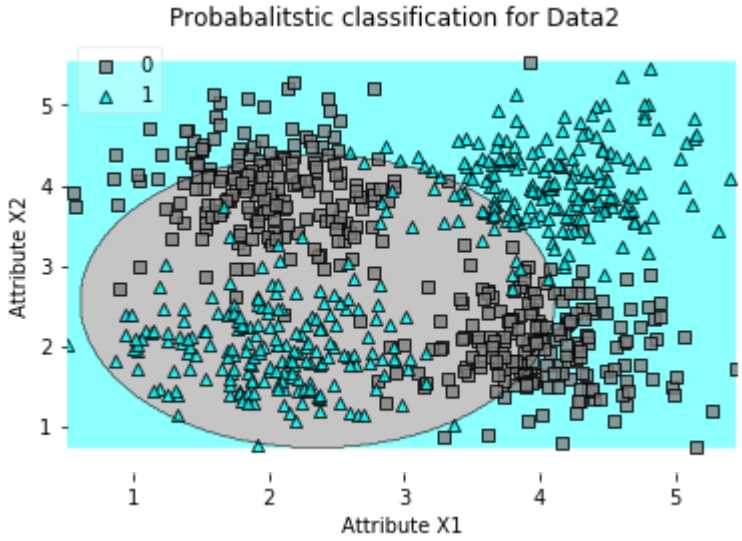
In [14]:

```
nb1 = GaussianNB()
nb1.fit(Data1_X, Data1_Y)
plot_decision_regions(Data1_X, Data1_Y, clf=nb1,colors='gray,cyan', legend=2,zoom_facto
r=100)
plt.xlabel('Attribute X1')
plt.ylabel('Attribute X2')
plt.title('Probabalitstic classification for Data2  ' )
plt.show()
```



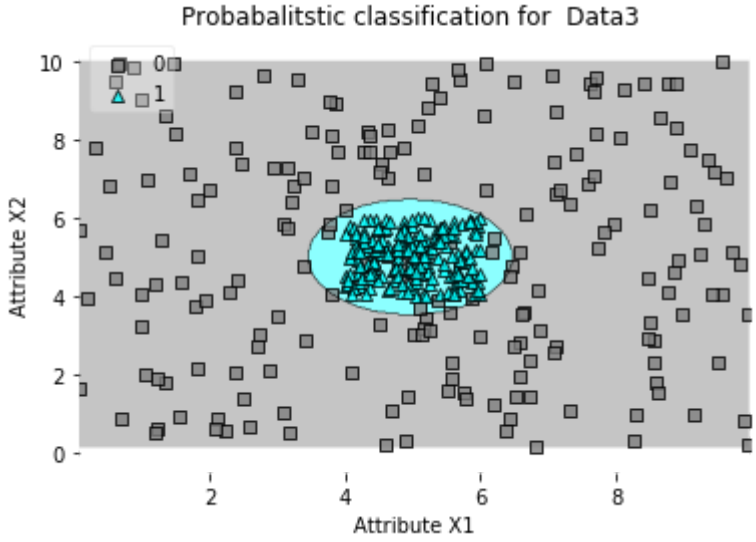
In [15]:

```
nb2 = GaussianNB()
nb2.fit(Data2_X, Data2_Y)
plot_decision_regions(Data2_X, Data2_Y, clf=nb2,colors='gray,cyan', legend=2,zoom_facto
r=100)
plt.xlabel('Attribute X1')
plt.ylabel('Attribute X2')
plt.title('Probabalitstic classification for Data2  ' )
plt.show()
```



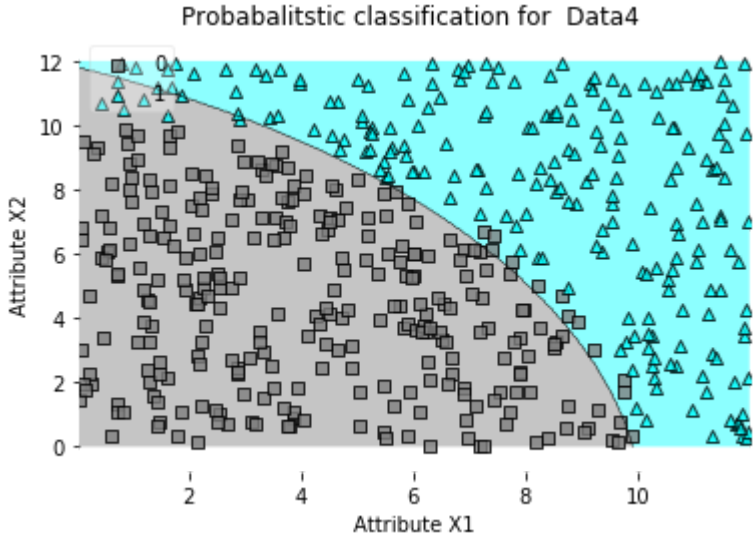
In [16]:

```
nb3 = GaussianNB()
nb3.fit(Data3_X, Data3_Y)
plot_decision_regions(Data3_X, Data3_Y, clf=nb3,colors='gray,cyan', legend=2,zoom_facto
r=100)
plt.xlabel('Attribute X1')
plt.ylabel('Attribute X2')
plt.title('Probabalitstic classification for  Data3 ' )
plt.show()
```



In [17]:

```
nb4 = GaussianNB()
nb4.fit(Data4_X, Data4_Y)
plot_decision_regions(Data4_X, Data4_Y, clf=nb4,colors='gray,cyan', legend=2,zoom_facto
r=100)
plt.xlabel('Attribute X1')
plt.ylabel('Attribute X2')
plt.title('Probabalitstic classification for  Data4 ' )
plt.show()
```



Question 3d: Describe the shape of the decision boundary on all four datasets. Explain the reason.

Answer: For data1 ,the decision boundary is a straight line(similar to linear SVM) and seperated the two classes linearly ,For Data2/data3 the decision boundary is ellipitical(like a probablistic model) and for data4 the decision boundary looks a polynomial function of degree 2(like Non linear SVM of poly kernel of degree 2).

In [4]:

```
print('Mean of Data1=' +str(np.mean(Data1_X,axis=0)))

print('Variance of Data1=' +str(np.var(Data1_X,axis=0)))

print('Mean of Data2=' +str(np.mean(Data2_X,axis=0)))

print('Variance of Dat2=' +str(np.var(Data2_X,axis=0)))

print('Mean of Data3=' +str(np.mean(Data3_X,axis=0)))

print('Variance of Data3=' +str(np.var(Data3_X,axis=0)))

print('Mean of Data4=' +str(np.mean(Data4_X,axis=0)))

print('Variance of Data4=' +str(np.var(Data4_X,axis=0)))
```

```
Mean of Data1=[3.00433106 2.95794583]
Variance of Data1=[1.5233972 1.4862435]
Mean of Data2=[3.02171742 2.9907841 ]
Variance of Dat2=[1.30545689 1.2410328 ]
Mean of Data3=[5.07096072 5.03790801]
Variance of Data3=[3.88470502 4.22884544]
Mean of Data4=[5.95013027 6.00621474]
Variance of Data4=[11.8798124  11.87709105]
```

Question 3e: Based on your plots in **Question 3c** explain the poor performance of NB on some datasets.

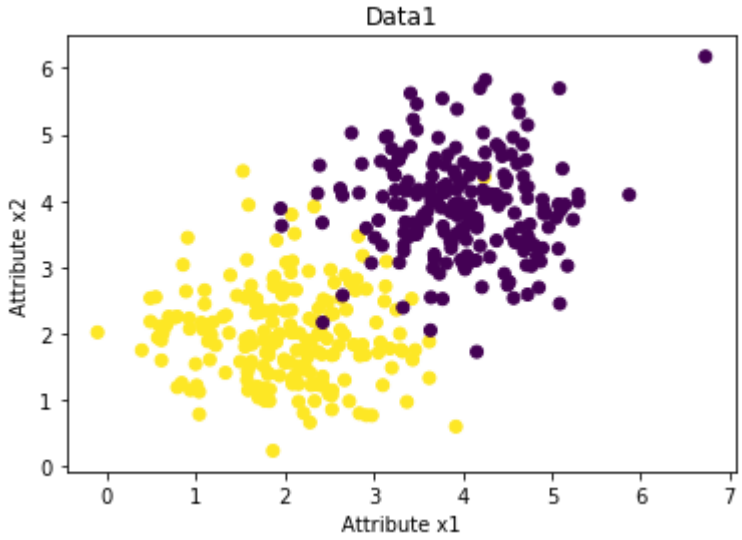
Answer: The poor performance has been obsereved for the Datasets2 when classified using NB classifier because the points were spread in such a way that the means of the classess became close to each other(almost equal) confusing the classification technique.

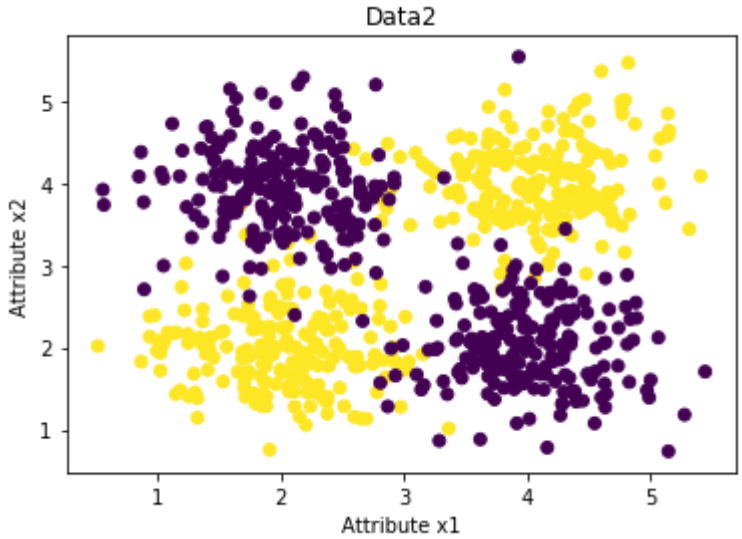
4. Support Vector Machines (Linear)

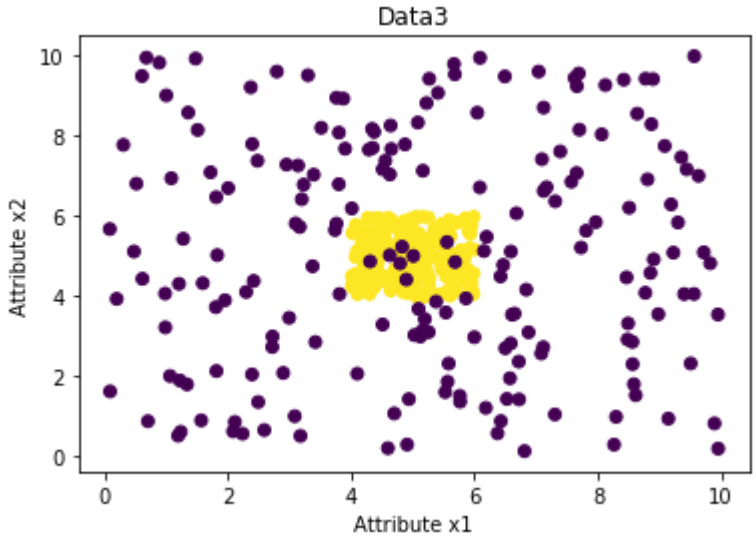
Question 4a: Based on the visualization of the four datasets, assess how well a linear SVM is expected to perform. Specifically, rank the datasets in the order of decreasing accuracy when a linear SVM is used. No need to compute accuracy to answer this question.

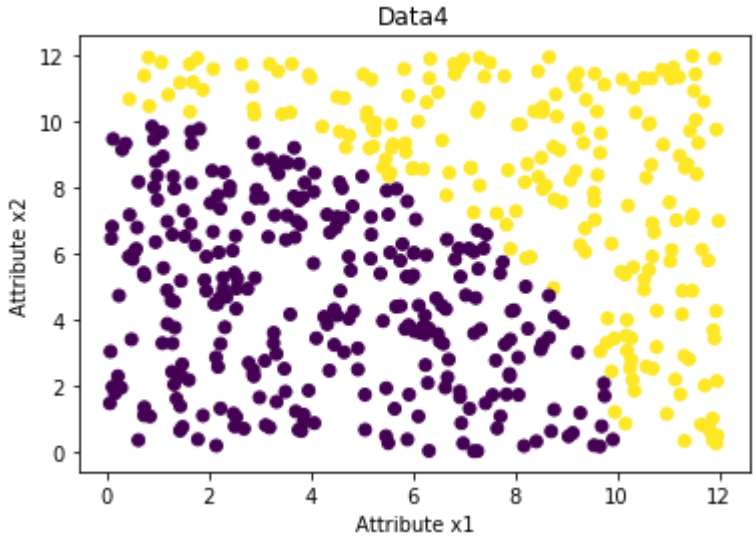
In [19]:

```
plt.scatter(Data1_X[:,0],Data1_X[:,1], c=Data1_Y)
plt.xlabel('Attribute x1')
plt.ylabel('Attribute x2')
plt.title('Data1')
plt.show()

plt.scatter(Data2_X[:,0],Data2_X[:,1], c=Data2_Y)
plt.xlabel('Attribute x1')
plt.ylabel('Attribute x2')
plt.title('Data2')
plt.show()

plt.scatter(Data3_X[:,0],Data3_X[:,1], c=Data3_Y)
plt.xlabel('Attribute x1')
plt.ylabel('Attribute x2')
plt.title('Data3')
plt.show()

plt.scatter(Data4_X[:,0],Data4_X[:,1], c=Data4_Y)
plt.xlabel('Attribute x1')
plt.ylabel('Attribute x2')
plt.title('Data4')
plt.show()
```


Data1
Attribute x2
Attribute x1
0
1
2
3
4
5
6
7


Data2
Attribute x2
Attribute x1
1
2
3
4
5





Answer: Decreasing order of accuracies were Data1,Data4,Data3,Data2

Question 4b: Compute and print the 10-fold cross-validation accuracy for a linear SVM classifier on all four datasets: Data1, Data2, Data3, Data4

In [20]:

```
computation_list=[[Data1_X, Data1_Y],[Data2_X, Data2_Y],[Data3_X, Data3_Y],[Data4_X, Da
ta4_Y]]

for i in range(4):
    for j in range(1):
        svm_linear = SVC(C=0.5, kernel='linear')
        svm_linear_scores = cross_val_score(svm_linear, computation_list[i][j], computa
tion_list[i][j+1], cv=10,scoring='accuracy')
        print("The Mean accuracy of the Data "+str(i+1)+" would be "+ str(svm_linear_sc
ores.mean()*100) )
```

```
The Mean accuracy of the Data 1 would be 96.74999999999999
The Mean accuracy of the Data 2 would be 14.125000000000002
The Mean accuracy of the Data 3 would be 64.25
The Mean accuracy of the Data 4 would be 92.59463785514207
```

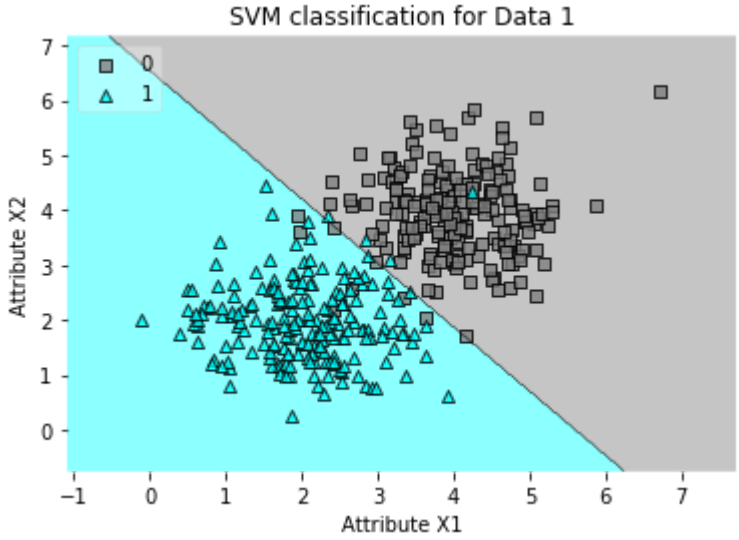
Question 4c: Rank the datasets in the decreasing order of accuracy of SVM.

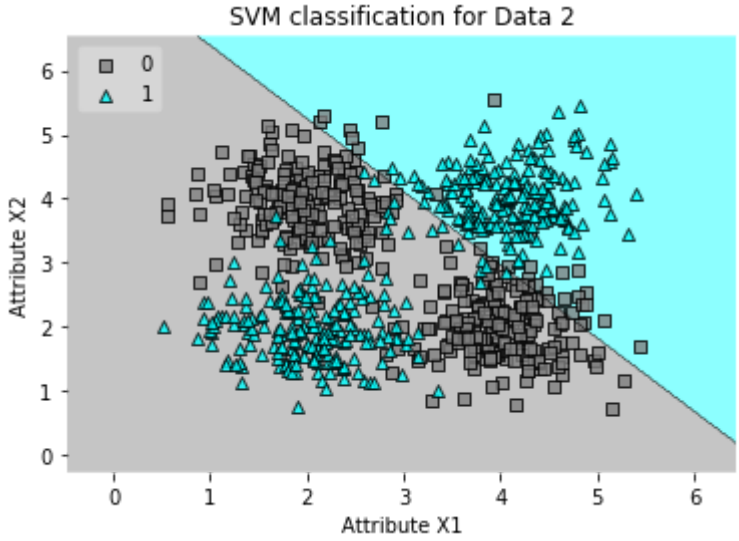
Answer: Data1,Data4,Data3,Data2 be the decreasing order of accuracies when classified using SVM classifier.

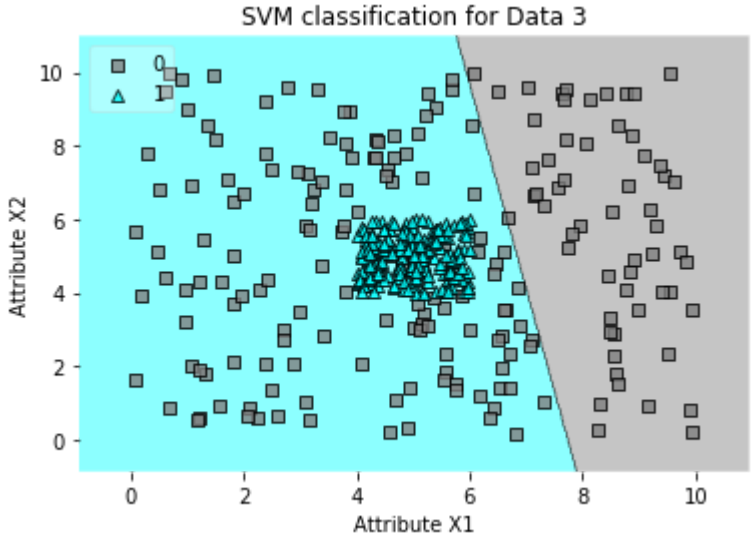
Question 4d: Plot decision regions for a linear SVM classifier on each of the four datasets

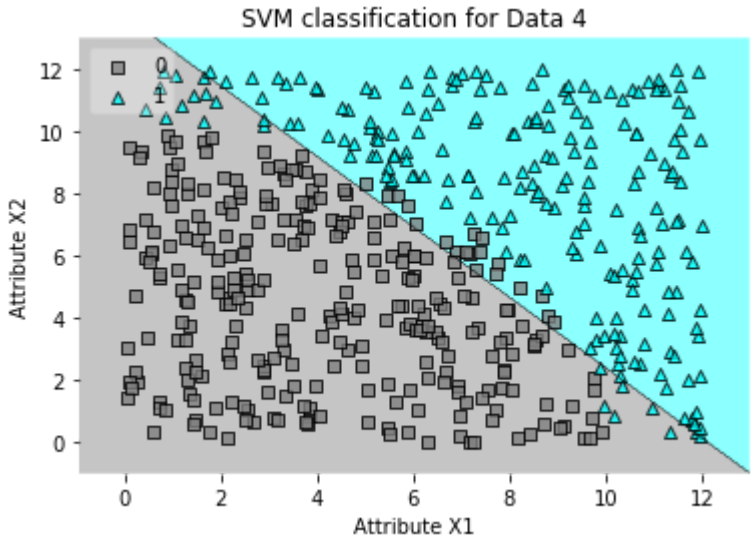
In [21]:

```
computation_list=[[Data1_X, Data1_Y],[Data2_X, Data2_Y],[Data3_X, Data3_Y],[Data4_X, Da
ta4_Y]]

for i in range(4):
    for j in range(1):
        svm_linear = SVC(C=0.5, kernel='linear')
        svm_linear.fit(computation_list[i][j], computation_list[i][j+1])
        plot_decision_regions(computation_list[i][j], computation_list[i][j+1], clf=svm
_linear,colors='gray,cyan', legend=2)
        plt.xlabel('Attribute X1')
        plt.ylabel('Attribute X2')
        plt.title('SVM classification for Data '+ str(i+1) )
        plt.show()
```


SVM classification for Data 1
0
1
Attribute X2
Attribute X1


SVM classification for Data 2
0
1
Attribute X2
Attribute X1


SVM classification for Data 3
0
1
Attribute X2
Attribute X1



Question 4e: Explain the reason for your observations in **Question 4c** using observations from the above decision regions.

Answer: The Objective of the svm is to find a optimal hyperplane that maximizes the margin and minimizes the sum of the slack of the points that crosses the margin of a particular class

For Data 1,the points were less noisy and only few points crossed the margin and sum of the slack of this points will be less and from the decion regions SVM classified the points correctly.

For Data 2,the points were very noisy and the points dispersed randomly, many points crossed the margin and sum of the slack of this points will be greater than one and it would be more , from the decion regions SVM find difficult to classify the points,so it has less accuracy.

For Data 3,the points were very noisy and many points crossed the margin and sum of the slack of this points will be greater than one and it would be more but less compared to sum of the slacks in data2 from the decion regions SVM find difficult to classify the points,so it has less accuracy but greater than data2.

For Data 4,the points were less noisy and only few points crossed the margin and sum of the slack of this points will be less and greater than 1 , from the decion regions SVM classified the points correctly but it has accuracy less when compared wth data1.

5. Non-linear Support Vector Machines

Use **Data2** to answer the following questions.

Question 5a: Compute and print the 10-fold cross-validation accuracy for an SVM with a polynomial kernel and degree values 1, 2, and 3.

In [22]:

```
degree=[1,2,3]
for i in degree:
    svm_poly = SVC(C=0.5, kernel='poly',degree=i, gamma = 'auto')
    svm_poly_scores = cross_val_score(svm_poly, Data2_X, Data2_Y, cv=10, scoring='accur
acy')
    print("The accuracy of the Non linear SVM classifier with kernel of type  poly ,deg
ree  "+ str(i) +" would be \n  " +  str(svm_poly_scores*100) )
```

```
The accuracy of the Non linear SVM classifier with kernel of type  poly ,d
egree  1 would be
  [13.75 12.5   1.25  8.75 17.5  18.75 10.   16.25 18.75 16.25]
The accuracy of the Non linear SVM classifier with kernel of type  poly ,d
egree  2 would be
  [81.25 83.75 88.75 83.75 88.75 88.75 86.25 88.75 91.25 83.75]
The accuracy of the Non linear SVM classifier with kernel of type  poly ,d
egree  3 would be
  [82.5  87.5  88.75 86.25 92.5  90.   86.25 88.75 88.75 85.  ]
```

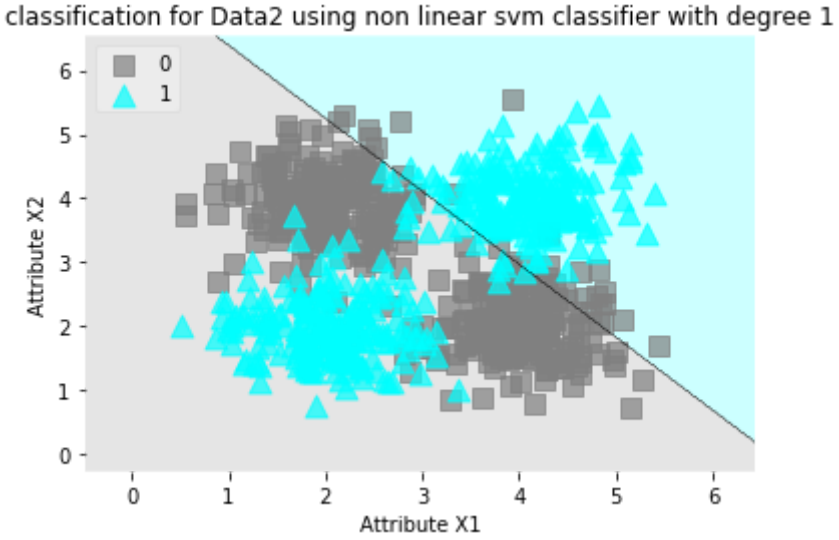
Question 5b: Rank the polynomial kernels in decreasing order of accuracy.

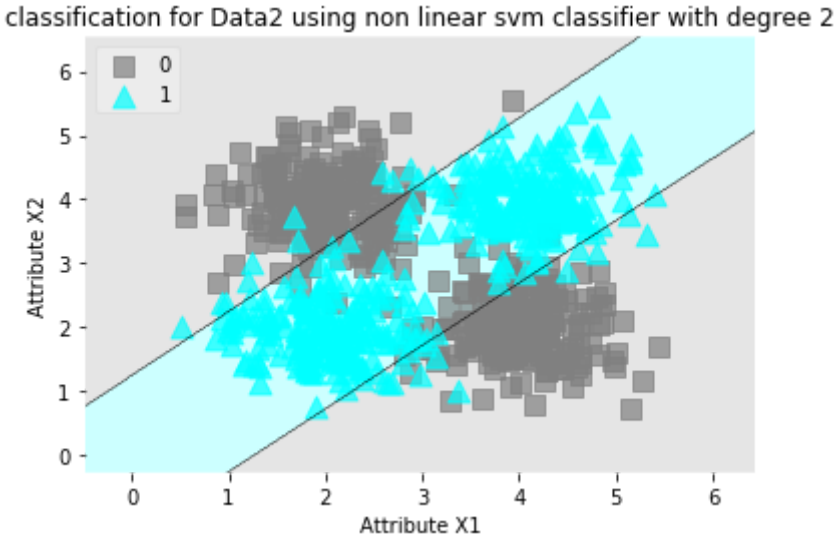
Answer: A nonlinear svm classifier with degree 3 had more accuracy when compared with degree 2 and degree 1 and decreasing order be nonlinear svm(degree=3),nonlinear svm(degree=2),nonlinear svm(degree=1).

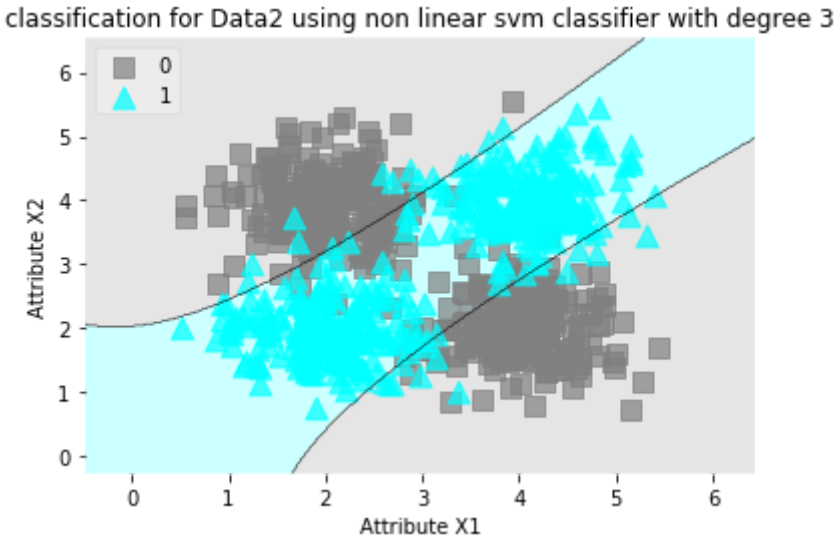
Question 5c: Plot decision regions for a polynomial kernel SVM with degree values 1, 2, and 3.

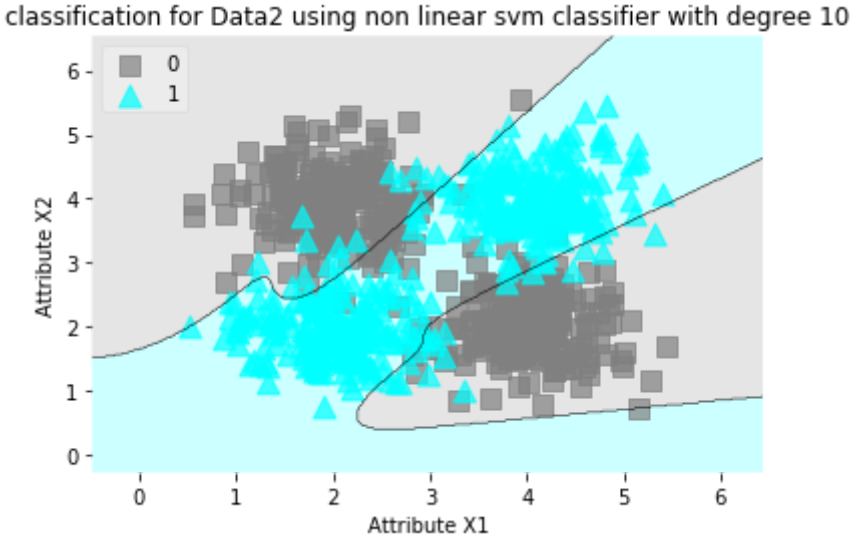
In [38]:

```
degree=[1,2,3,10]
scatter_kwargs = {'s': 120, 'edgecolor': None, 'alpha': 0.7}
contourf_kwargs = {'alpha': 0.2}
scatter_highlight_kwargs = {'s': 120, 'label': 'Test data', 'alpha': 0.7}
for i in degree:
    svm_poly = SVC(C=0.5, kernel='poly',degree=i, gamma = 'auto')
    svm_poly.fit(Data2_X, Data2_Y)
    plot_decision_regions(Data2_X, Data2_Y, clf=svm_poly,colors='gray,cyan',scatter_kwa
rgs=scatter_kwargs,
                      contourf_kwargs=contourf_kwargs,
                      scatter_highlight_kwargs=scatter_highlight_kwargs, legend=2)
    plt.xlabel('Attribute X1')
    plt.ylabel('Attribute X2')
    plt.title('classification for Data2 using non linear svm classifier with degree '+
str(i) )
    plt.show()
```


classification for Data2 using non linear svm classifier with degree 1
0
1
Attribute X2
Attribute X1
0
1
2
3
4
5
6


classification for Data2 using non linear svm classifier with degree 2
0
1
Attribute X2
Attribute X1
0
1
2
3
4
5
6





Question 5d: Based on the decision regions, explain the reason for your observations in **Question 5c**.

Answer: From the Decision regions a non linear SVM classifier with degree 3 seperated the classes effectively when comapred with Degree 2 and Degree 1 as its decision boundary classified few additional points which were misclassified when using degree 2 and degree1 .so it has good accuracy when compared with degree 2 and degree 1 and in sync with my observation in 5b.

Question 5e: Compute the 10-fold cross-validation accuracy for an SVM with an RBF kernel and gamma values 0.01, 0.1, and 1.

In [24]:

```
gamma=[0.01,0.1,1]
for x in gamma:
    svm_rbf = SVC(C = 0.5, kernel='rbf', gamma=x)
    svm_rbf_scores = cross_val_score(svm_rbf, Data2_X, Data2_Y, cv=10, scoring='accurac
y')
    print("The accuracy of a non linear SVM classifier with RBF kernel with gamma value
" + str(x) + " would be \n" +str(svm_rbf_scores)+' and the mean accuracy is  ' +str(svm
_rbf_scores.mean()*100)+' .')
```

```
The accuracy of a non linear SVM classifier with RBF kernel with gamma val
ue 0.01 would be
[0.375  0.3125 0.0875 0.25   0.4375 0.3375 0.3    0.3    0.275  0.3375] an
d the mean accuracy is  30.124999999999996 .
The accuracy of a non linear SVM classifier with RBF kernel with gamma val
ue 0.1 would be
[0.975  0.9    0.9375 0.9    0.9625 0.9375 0.8875 0.9375 0.9625 0.9625] an
d the mean accuracy is  93.625 .
The accuracy of a non linear SVM classifier with RBF kernel with gamma val
ue 1 would be
[0.9875 0.9125 0.95   0.925  0.9625 0.9375 0.875  0.9375 0.9625 0.95  ] an
d the mean accuracy is  93.99999999999999 .
```

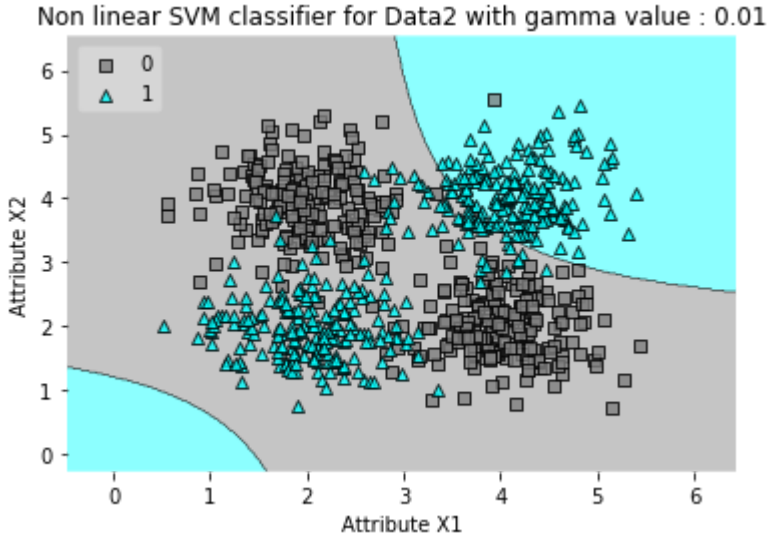
Question 5f: Rank the RBF kernels in decreasing order of accuracy.

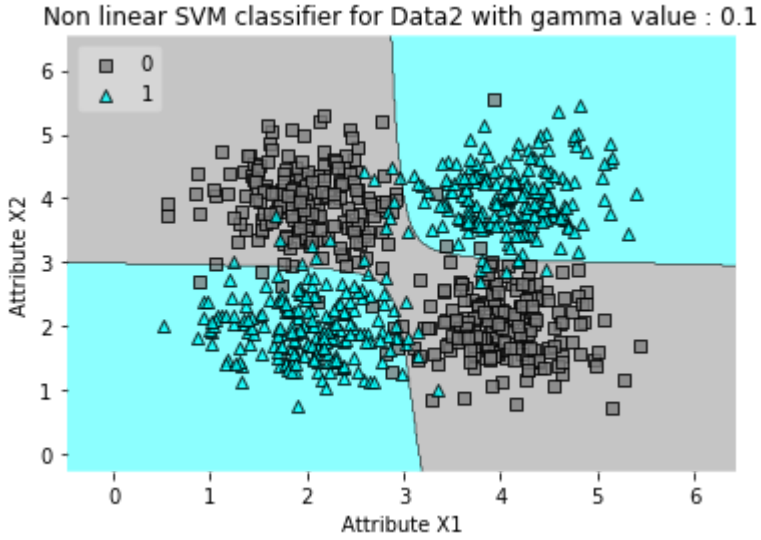
Answer: RBF kernel with gamma=1,RBF kernel with gamma=0.1 and RBF kernel with gamma=0.01 will be decreasing order of accuracy.

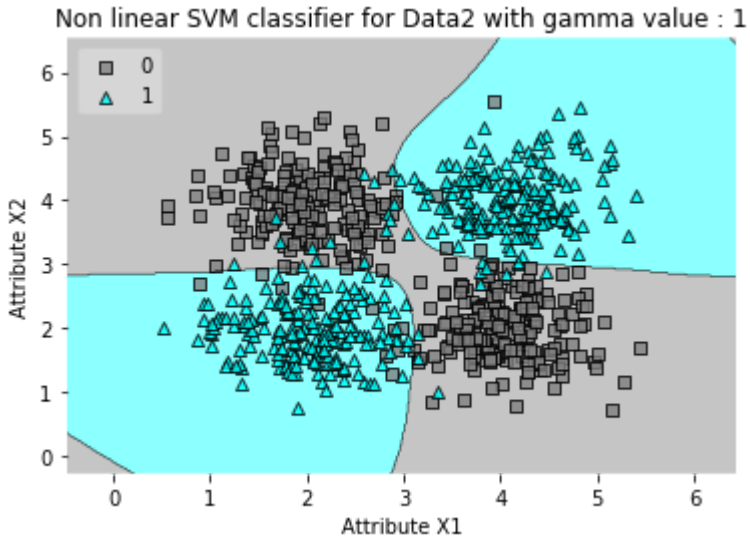
Question 5g: Plot decision regions for the above RBF Kernels

In [25]:

```
gamma=[0.01,0.1,1]
for x in gamma:
    svm_rbf = SVC(C = 0.5, kernel='rbf', gamma=x)
    svm_rbf.fit(Data2_X,Data2_Y)
    plot_decision_regions(Data2_X, Data2_Y, clf=svm_rbf,colors='gray,cyan', legend=2)
    plt.xlabel('Attribute X1')
    plt.ylabel('Attribute X2')
    plt.title('Non linear SVM classifier for Data2 with gamma value : ' + str(x)  )
    plt.show()
```


Non linear SVM classifier for Data2 with gamma value : 0.01
0
1
Attribute X2
Attribute X1
0
1
2
3
4
5
6


Non linear SVM classifier for Data2 with gamma value : 0.1
0
1
Attribute X2
Attribute X1
0
1
2
3
4
5
6



Question 5h: Explain the reason for your observations in **Question 5f** from the above decision regions.

Answer: In the radial basis function increasing gamma values makes the bell shape curve naroower and as a result each instances range of influence is smaller,when gamma =0.01 the bell curve/decision region is wider and it didn't classified the points correctly and this descision region is not suited for this data.

When gamma=0.1 the bell curve becomes narrower when compared with gamma=0.01 and the decision region becames narrower and it classified more points correctly when comapared with gamma 0.01 and as result accuracy is increased for this model.

when gamma=1,the decision region became more narrower and suited to this data perfectly and classified the points correctly.so this model has better accuracy when compared with model(gamma=0.01 and gamma=0.1) and in sync with my observation in 5f.

Question 5i: Between SVM with a Polynomial kernel and SVM with an RBF kernel, which one is ideally suited of Data3? Explain your reason.

Answer: RBF Kernel will be better suited for Data2 when compared with polynomial kernel because when gamma=1 the bell narrow shaped curve acts like a perfect decision region to seperate classes with out any overlap.

6. Classification Evaluation

Question 6a:

Run SVM classifier (with RBF kernel and gamma=0.1) on **Data2** and compute the mean of k-fold cross-validation accuracies for cv = 3, 4, 5 and 6. Report the mean of accuracies for each choice of 'cv' and explain the reason for any differences in the mean accuracy you observe.

In [37]:

```
#cv=[3,4,5,6,10,25,50,700]
cv=[3,4,5,6]
for k in cv:
    svm_rbf = SVC(C = 0.5, kernel='rbf', gamma=0.1)
    svm_rbf_scores = cross_val_score(svm_rbf, Data2_X, Data2_Y, cv=k, scoring='accurac
y')
    print('The accuracy of a non linear SVM classifier with RBF kernel with gamma value
0.1 and for the cross folds ' + str(k) + '\n and the mean accuracy is '+str((svm_rbf_sc
ores.mean())*100)+' .')
```

```
The accuracy of a non linear SVM classifier with RBF kernel with gamma val
ue 0.1 and for the cross folds 3
 and the mean accuracy is 90.3826731006621 .
The accuracy of a non linear SVM classifier with RBF kernel with gamma val
ue 0.1 and for the cross folds 4
 and the mean accuracy is 91.625 .
The accuracy of a non linear SVM classifier with RBF kernel with gamma val
ue 0.1 and for the cross folds 5
 and the mean accuracy is 92.75 .
The accuracy of a non linear SVM classifier with RBF kernel with gamma val
ue 0.1 and for the cross folds 6
 and the mean accuracy is 93.25531433740389 .
```

Answer: when k=3 ,the data is divided into 3 chunks and in each of the 3 iterations each chunk will act as a test data and the remaining two chunks will act as training data and in K cross validation each data point in the dataset will be tested only once and each data point involved in (k-1) training phases,here it is two.when the cross fold value increases the model will train more number of times i.e by more combinations of training datasets which provides some inferences like in which instance the model performed better which results in differences in the accuracies/mean accuracies.

Question 6b:

For DT, NB, kNN, Linear SVM, Polynomial Kernel SVM, and SVM with RBF kernel classifiers, compute the 30-fold crossvalidation **accuracies** and **precision** (use scoring='precision' when calling cross_val_score()) on **Data3**. Rank the classifiers based on accuracy and precision scores. Are the best classifiers ranked according to accuracy and precision the same? If not, explain the reason.

For the classifiers, feel free to choose any parameter settings you prefer.

In [5]:

```
dt = DecisionTreeClassifier(max_depth=4)
dt_scores_accuracy = cross_val_score(dt, Data3_X, Data3_Y, cv=30, scoring='accuracy')
dt_scores_precision = cross_val_score(dt, Data3_X, Data3_Y, cv=30, scoring='precision')
print("The accuracy of the decision tree classifier with max_depth =4 would be " +str(d
t_scores_accuracy.mean()*100) )
print("The precision of the decision tree classifier with max_depth =4 would be " +str(
dt_scores_precision.mean()*100) )
nb=GaussianNB()
nb_scores_accuracy = cross_val_score(nb, Data3_X, Data3_Y, cv=30, scoring='accuracy')
nb_scores_precision = cross_val_score(nb, Data3_X, Data3_Y, cv=30, scoring='precision')
print("The accuracy of the naive bayes classifier  would be " +str(nb_scores_accuracy.m
ean()*100) )
print("The precision of the naive bayes classifier would be " +str(nb_scores_precision.
mean()*100) )
KNN= KNeighborsClassifier(n_neighbors=5)
KNN_scores_accuracy = cross_val_score(KNN, Data3_X, Data3_Y, cv=30, scoring='accuracy')
KNN_scores_precision = cross_val_score(KNN, Data3_X, Data3_Y, cv=30, scoring='precisio
n')
print("The accuracy of the K-Neighbors Classifier  would be " +str(KNN_scores_accuracy.
mean()*100) )
print("The precision of the K-Neighbors Classifier would be " +str(KNN_scores_precision
.mean()*100) )
svm_linear=SVC(C=0.5,kernel='linear')
svm_linear_accuracy = cross_val_score(svm_linear, Data3_X, Data3_Y, cv=30, scoring='acc
uracy')
svm_linear_precision = cross_val_score(svm_linear, Data3_X, Data3_Y, cv=30, scoring='pr
ecision')
print("The accuracy of the SVM linear Classifier  would be " +str(svm_linear_accuracy.m
ean()*100) )
print("The precision of the SVM linear Classifier would be " +str(svm_linear_precision.
mean()*100) )
svm_poly=SVC(C=0.5,kernel='poly',degree=2,gamma='auto')
svm_poly_accuracy = cross_val_score(svm_poly, Data3_X, Data3_Y, cv=30, scoring='accurac
y')
svm_poly_precision = cross_val_score(svm_poly, Data3_X, Data3_Y, cv=30, scoring='precis
ion')
print("The accuracy of the Non linear SVM Classifier with  kernel of polynomial 2 would
be " +str(svm_poly_accuracy.mean()*100) )
print("The precision of the Non linear SVM Classifier with  kernel of polynomial 2 woul
d be " +str(svm_poly_precision.mean()*100) )

svm_rbf=SVC(C=0.5,kernel='rbf',gamma=1)
svm_rbf_accuracy = cross_val_score(svm_rbf, Data3_X, Data3_Y, cv=30, scoring='accuracy'
)
svm_rbf_precision = cross_val_score(svm_rbf, Data3_X, Data3_Y, cv=30, scoring='precisio
n')
print("The accuracy of the Non linear SVM Classifier with rbf kernel and gamma value on
e would be " +str(svm_rbf_accuracy.mean()*100) )
print("The precision of the Non linear SVM Classifier with rbf kernel and gamma value o
ne  would be " +str(svm_rbf_precision.mean()*100) )
```

```
The accuracy of the decision tree classifier with max_depth =4 would be 9
7.18253968253968
The precision of the decision tree classifier with max_depth =4 would be 9
6.04497354497356
The accuracy of the naive bayes classifier  would be 95.91269841269843
The precision of the naive bayes classifier would be 93.28571428571429
The accuracy of the K-Neighbors Classifier  would be 94.92063492063492
The precision of the K-Neighbors Classifier would be 91.88311688311688
The accuracy of the SVM linear Classifier  would be 64.28571428571429
The precision of the SVM linear Classifier would be 58.817016317016325
The accuracy of the Non linear SVM Classifier with  kernel of polynomial 2
would be 84.84126984126983
The precision of the Non linear SVM Classifier with  kernel of polynomial
2 would be 79.23713323713325
The accuracy of the Non linear SVM Classifier with rbf kernel and gamma va
lue one would be 95.43650793650794
The precision of the Non linear SVM Classifier with rbf kernel and gamma v
alue one  would be 92.65692640692642
```

Answer: Based on accuracy below is the ranking : 1)Decision tree 2)Naive Bayes 3)SVM Classifier with rbf kernel and gamma value one 4)K-neighbours, 5)SVM Classifier with kernel of polynomial 2, 6)Linear SVM classifier.

Based on precision below is the ranking: 1)Decision tree 2)Naive Bayes 3)SVM Classifier with rbf kernel and gamma value one 4)K-neighbours 5)SVM Classifier with kernel of polynomial 2, 6)Linear SVM classifier.

Yes the Best classifiers when ranked using metrics Accauracy ,Precision are the same and it is Decision tree.

7. Ensemble Methods

Question 7a: **Bagging:** Create bagging classifiers each with n_estimators = 1,2,3,4,5,10, and 20. Use a **linear SVM** (with C=0.5) as a base classifier. Using **Data3**, compute the mean **5-fold** cross validation accuracies and standard deviation for each of the bagging classifiers. State your observations on how bagging affected the mean and standard deviation of the base classifier. Explain your reason for what may have lead to these observations.

In [13]:

```
SVM_linear = SVC(C=0.5,kernel='linear');
n_est_list = [1,2,3,4,5,10,20]
for n_est in n_est_list:
    # creating an instance of bagging classifier with 'n_est' estimators with base as s
vm_linear
    bagging = BaggingClassifier(base_estimator=SVM_linear, n_estimators=n_est)
    # compute cross-validation accuracy for each bagging classifier
    scores = cross_val_score(bagging, Data3_X, Data3_Y, cv=5, scoring='accuracy')
    print("Bagging Accuracy: %.2f (+/- %.2f) #estimators: %d" % (scores.mean(), scores.
std(), n_est))
```

```
Bagging Accuracy: 0.61 (+/- 0.06) #estimators: 1
Bagging Accuracy: 0.70 (+/- 0.05) #estimators: 2
Bagging Accuracy: 0.64 (+/- 0.03) #estimators: 3
Bagging Accuracy: 0.64 (+/- 0.06) #estimators: 4
Bagging Accuracy: 0.68 (+/- 0.06) #estimators: 5
Bagging Accuracy: 0.68 (+/- 0.09) #estimators: 10
Bagging Accuracy: 0.68 (+/- 0.03) #estimators: 20
```

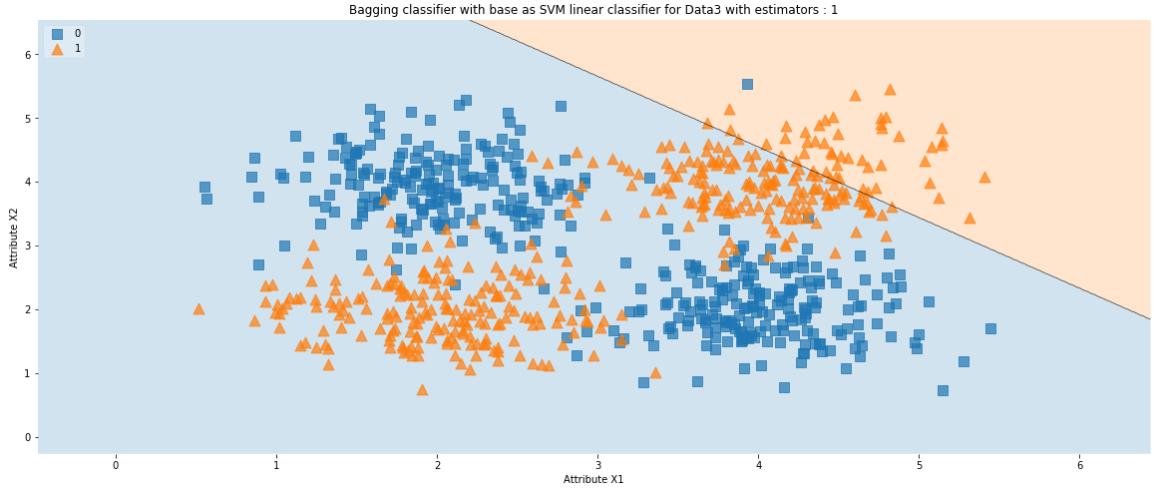
Answer: The mean accuracy of the base classifier has been increased and standard deviation/Variance of the base classifier is decreased as we are started increasing the estimators. From the above output when number of estimators =4 and number of estimators =10 the accuracy has been increased from 64 to 68 % and standard deviation has decreased from 6% to 3%.

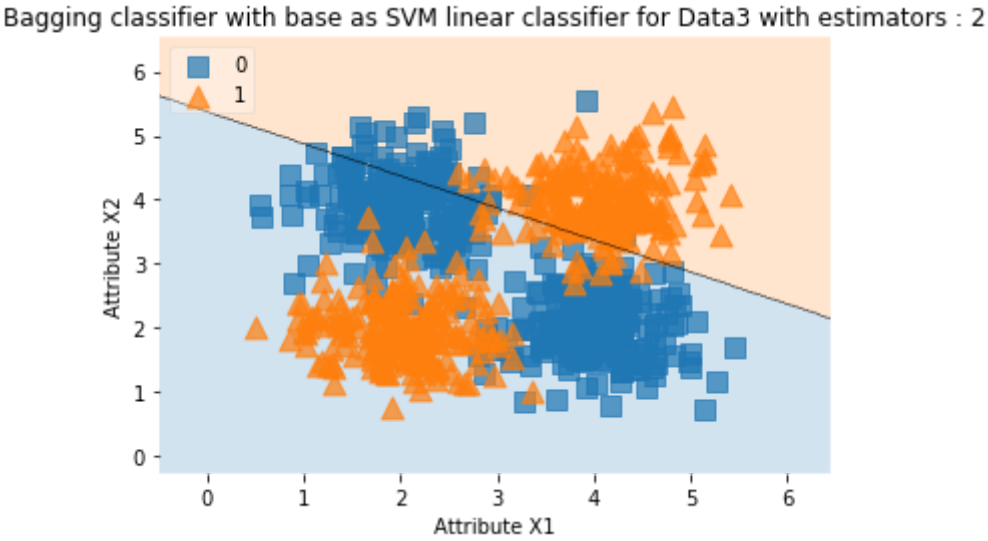
In bagging ,after all the predicators were trained the ensemble can make a prediction for a new instance (test point) by aggregating the predictions of all predicators and this aggregation may be a statistical mode ,the net result of the ensemble has a similar bias but a lower variance than a single base classifier because of this standard deviation is decreased ,mean is increased.

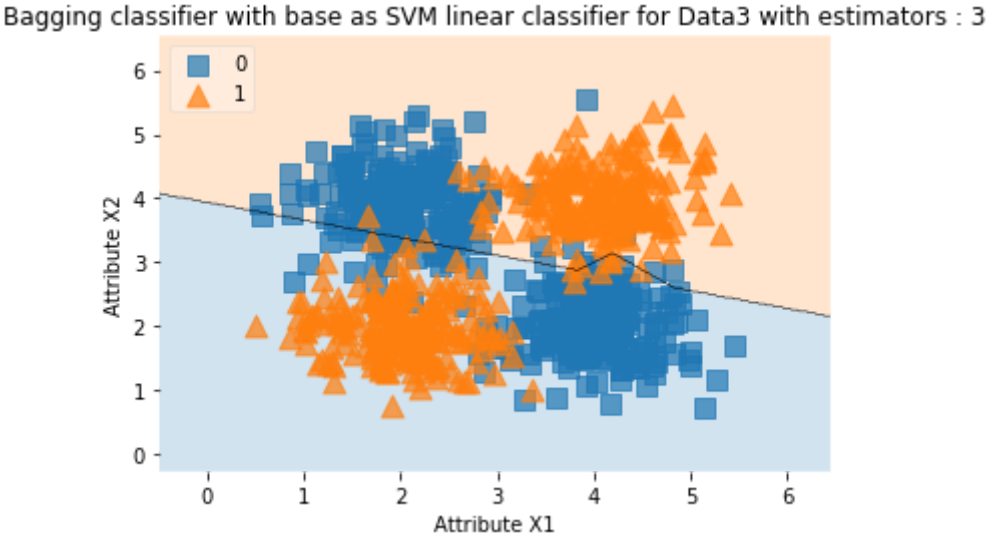
Question 7b: Plot decision regions for the above bagging classifiers.

In [12]:

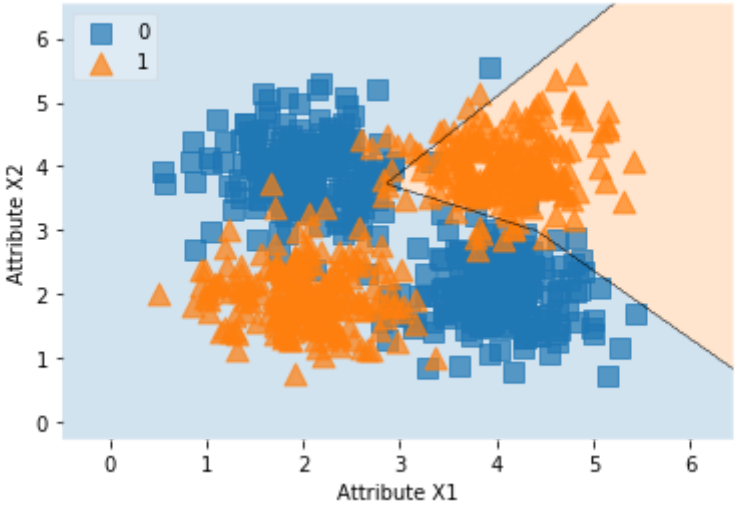
```
fig = plt.figure(figsize=(20, 8))
count = 0;
SVM_linear = SVC(C=0.5,kernel='linear');
n_est_list = [1,2,3,4,5,10,20]
scatter_kwargs = {'s': 120, 'edgecolor': None, 'alpha': 0.7}
contourf_kwargs = {'alpha': 0.2}
scatter_highlight_kwargs = {'s': 120, 'label': 'Test data', 'alpha': 0.7}
for n_est in n_est_list:
    count=count+1
    # creating an instance of bagging classifier with 'n_est' estimators with base as s
vm_linear
    bagging = BaggingClassifier(base_estimator=SVM_linear, n_estimators=n_est)
    bagging.fit(Data3_X,Data3_Y)
    plot_decision_regions(X=Data3_X, y=Data3_Y, clf=bagging, legend=2,
                          scatter_kwargs=scatter_kwargs,
                          contourf_kwargs=contourf_kwargs,
                          scatter_highlight_kwargs=scatter_highlight_kwargs)
    plt.xlabel('Attribute X1')
    plt.ylabel('Attribute X2')
    plt.title('Bagging classifier with base as SVM linear classifier for Data3 with est
imators : ' + str(n_est)  )
    plt.show()
```


Bagging classifier with base as SVM linear classifier for Data3 with estimators : 1
0
1
Attribute X2
Attribute X1

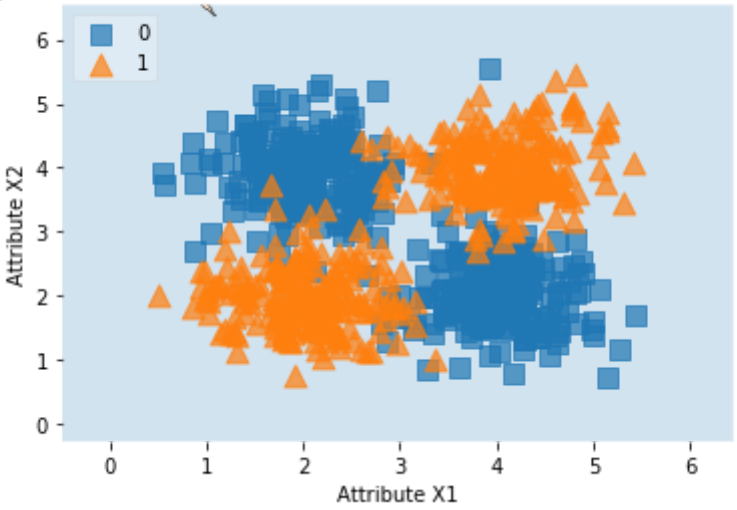

Bagging classifier with base as SVM linear classifier for Data3 with estimators : 2
0
1
Attribute X2
Attribute X1


Bagging classifier with base as SVM linear classifier for Data3 with estimators : 3
0
1
Attribute X2
Attribute X1

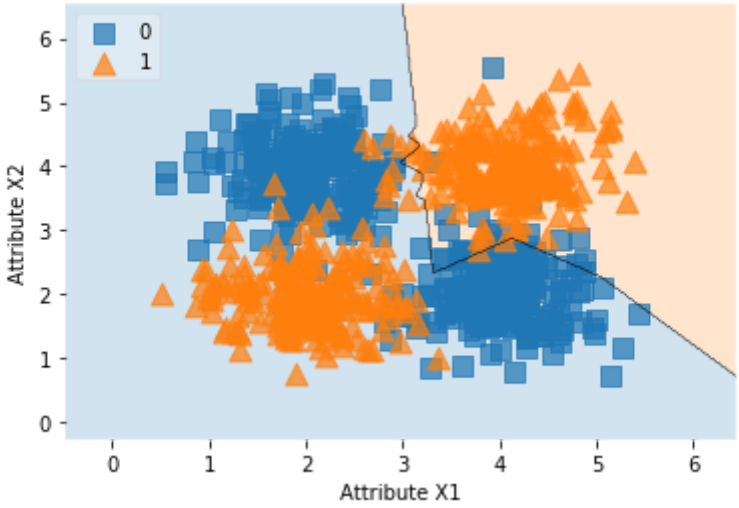
Bagging classifier with base as SVM linear classifier for Data3 with estimators : 4

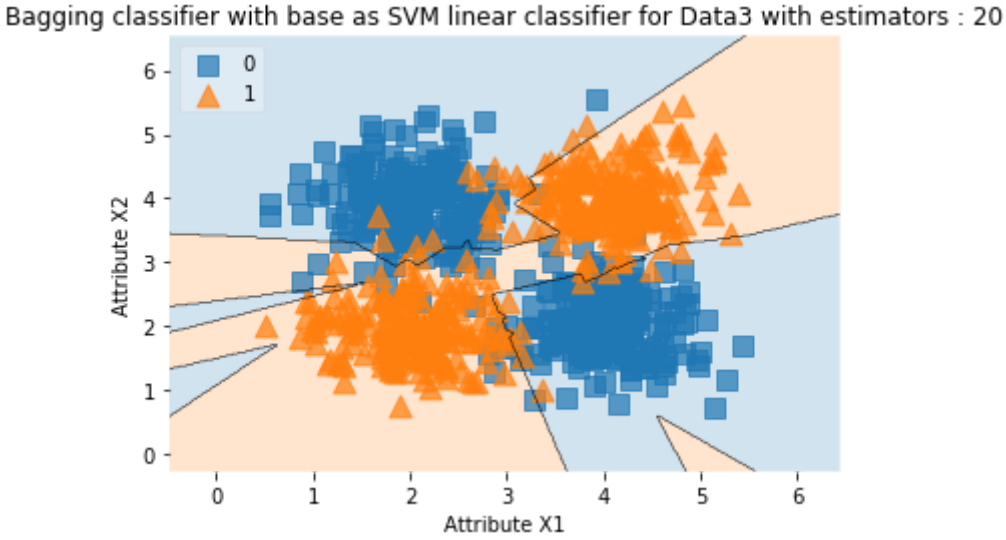


Bagging classifier with base as SVM linear classifier for Data3 with estimators : 5



Bagging classifier with base as SVM linear classifier for Data3 with estimators : 10





Question 7c: Comment on the quality of the decision regions for a bagging classifiers with many estimators when compared to that of only one estimator.

Answer: when we use bagging the ensemble produces an output with less variance when compared with a single base estimator. when the ensemble classifier has less variance when compared with single base estimator the ensemble moving downwards near to the tradeoff point in the overfitting and under fitting curve by finding the more generalization model,so the decision regions will be more generalized when compared with one estimator which is nothing but quality of the decision region increases.

Question 7d: **Boosting:** Create boosting classifiers each with n_estimators = 1,2,3,4,5,10, 20, and 40. Use a **Decision Tree** (with max_depth=2) as a base classifier. Using **Data2**, compute the mean **10-fold** cross validation accuracies and standard deviation for each of the bagging classifiers. State your observations on how boosting affected the mean and standard deviation of the base classifier.

In [14]:

```
dt = DecisionTreeClassifier(max_depth=2)
n_est_list = [1,2,3,4,5,10,20,40]
for n_est in n_est_list:
    # create an instance of a boosting classifier with 'n_est' estimators
    boosting = AdaBoostClassifier(base_estimator=dt, n_estimators=n_est)
    # compute cross-validation accuracy for each bagging classifier
    scores = cross_val_score(boosting, Data2_X, Data2_Y, cv=10, scoring='accuracy')
    print("Boosting Accuracy: %.2f (+/- %.2f) #estimators: %d" % (scores.mean(), scores
.std(), n_est))
```

```
Boosting Accuracy: 0.88 (+/- 0.03) #estimators: 1
Boosting Accuracy: 0.88 (+/- 0.03) #estimators: 2
Boosting Accuracy: 0.90 (+/- 0.04) #estimators: 3
Boosting Accuracy: 0.90 (+/- 0.04) #estimators: 4
Boosting Accuracy: 0.92 (+/- 0.03) #estimators: 5
Boosting Accuracy: 0.92 (+/- 0.04) #estimators: 10
Boosting Accuracy: 0.91 (+/- 0.04) #estimators: 20
Boosting Accuracy: 0.91 (+/- 0.02) #estimators: 40
```

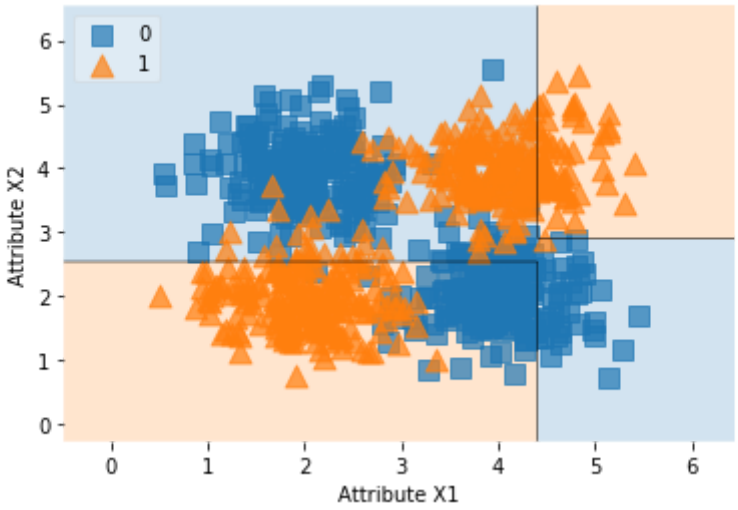
Answer: In boosting predicators are trained sequentially,each predictior is trying to correct its predecessor ,when estimators were increasing the mean value is increasing and the standard deviation is getting decreased ,when estimator =1 the ensemble has an accuracy of 88% of std of 3% and when estimators increased to 40 the mean accuracy became 91% and standard deviation becomes 2%.

Question 7e: Plot decision regions for above boosting classifiers. Explain your reason for what may have lead to the observations in **Question 7d**.

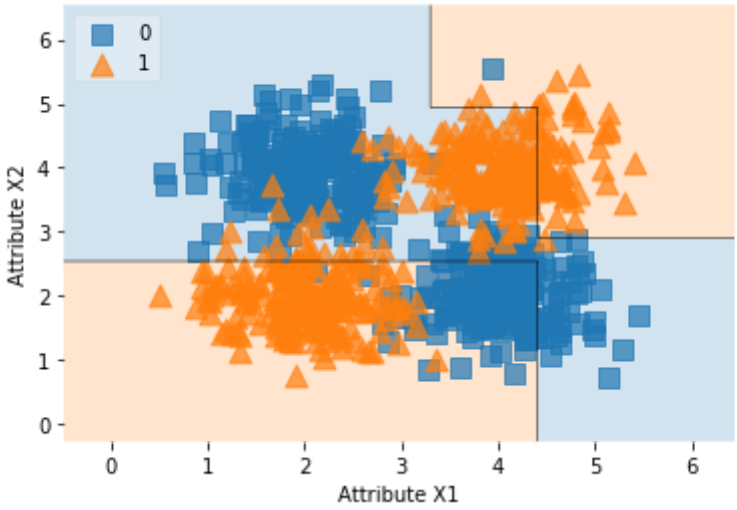
In [15]:

```
dt = DecisionTreeClassifier(max_depth=2)
n_est_list = [1,2,3,4,5,10,20,40]
for n_est in n_est_list:
    # create an instance of a boosting classifier with 'n_est' estimators
    boosting = AdaBoostClassifier(base_estimator=dt, n_estimators=n_est)
    boosting.fit(Data2_X,Data2_Y)
    plot_decision_regions(X=Data2_X, y=Data2_Y, clf=boosting, legend=2,
                          scatter_kwargs=scatter_kwargs,
                          contourf_kwargs=contourf_kwargs,
                          scatter_highlight_kwargs=scatter_highlight_kwargs)
    plt.xlabel('Attribute X1')
    plt.ylabel('Attribute X2')
    plt.title('Boosting classifier with base as Decision tree classifier for Data2 with
estimators : ' + str(n_est)  )
    plt.show()
```

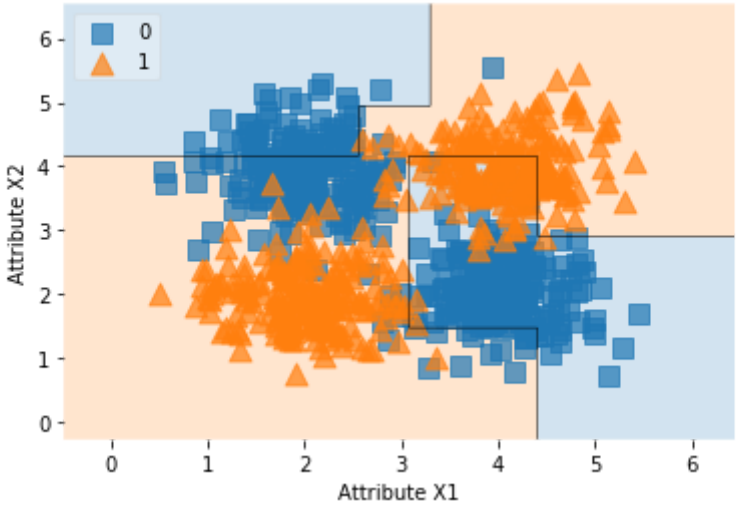
Boosting classifier with base as Decision tree classifier for Data2 with estimators : 1



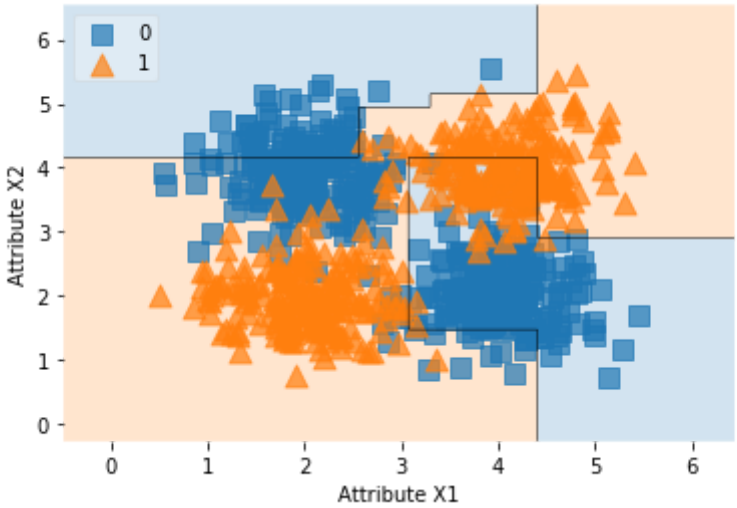
Boosting classifier with base as Decision tree classifier for Data2 with estimators : 2



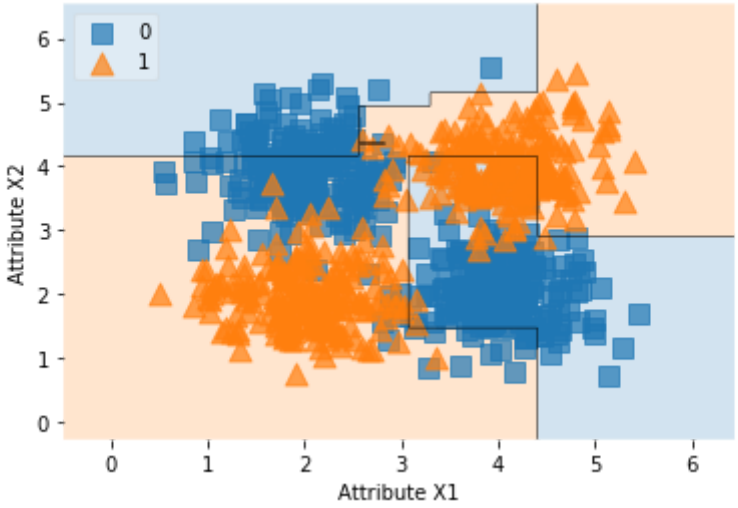
Boosting classifier with base as Decision tree classifier for Data2 with estimators : 3



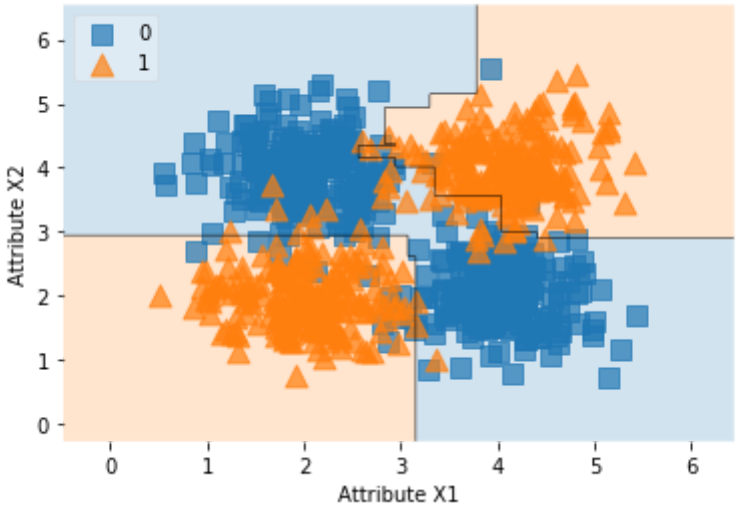
Boosting classifier with base as Decision tree classifier for Data2 with estimators : 4

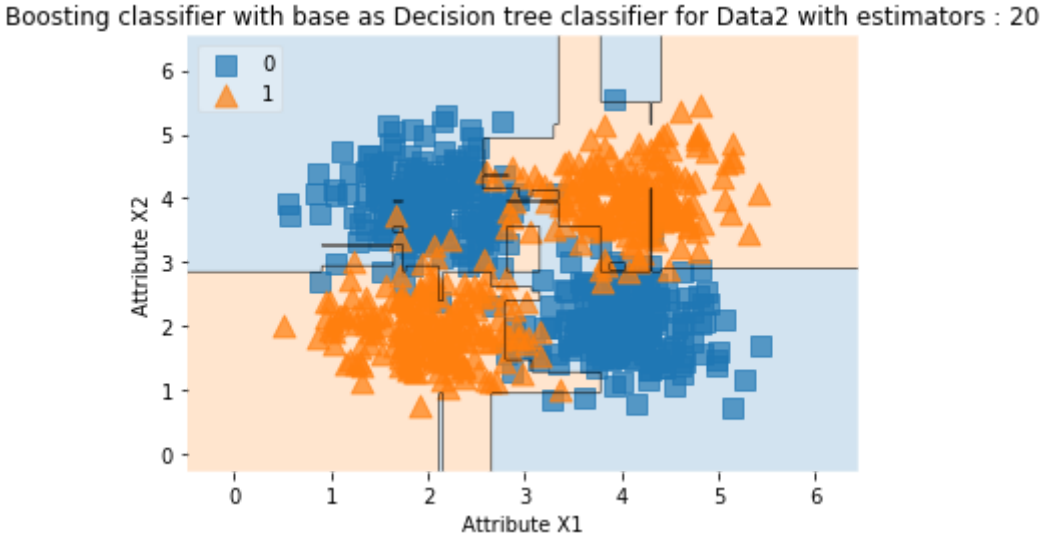


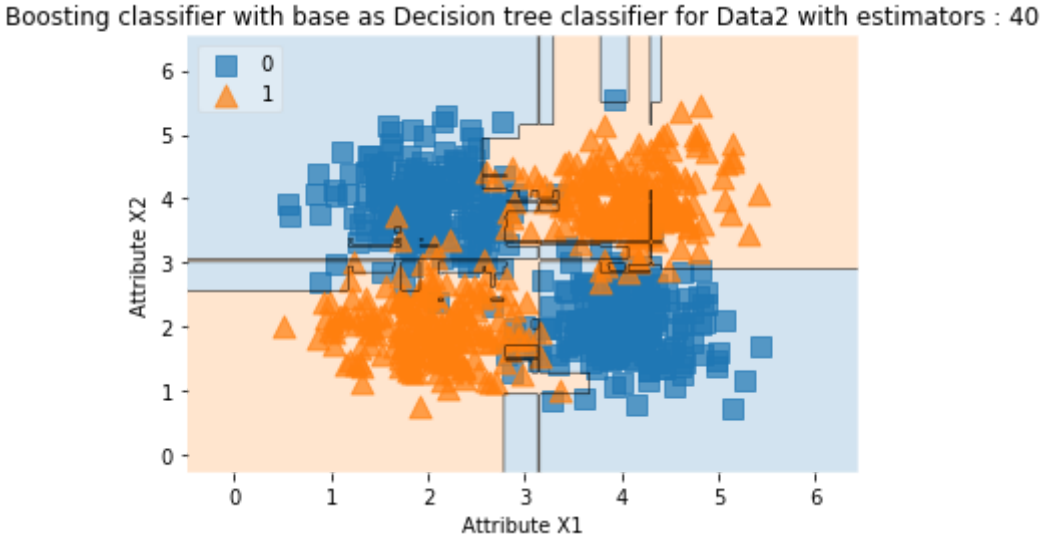
Boosting classifier with base as Decision tree classifier for Data2 with estimators : 5



Boosting classifier with base as Decision tree classifier for Data2 with estimators : 10







Answer: when the estimator values goes on increasing the decision boundaries were getting adjusted in such a way to correctly predict the previously misclassified points . when estimator =1 the descion boundary is from attributex1 <4.5 and attributex2 <2.5 and estimator=2 the decision boundary are getting expanded in such a way to included the misclassified point when using estimator as one.

when estimator =10 the decision boundary has been adjusted in such way to correctly classify the misclassified points with estimators less than 10.

In boosting we are making the we are combining the weak learners to strong learner model.

8. Classification on a real-world dataset

Real world datasets typically have many attributes making it hard to visualize. This question is about using SVM and Decision Tree algorithms on a real world 'breast cancer' dataset.

The following code reads the dataset from the 'datasets' library in sklearn.

In [5]:

```
from sklearn import datasets
cancer = datasets.load_breast_cancer()
```

The features are:

In [6]:

```
cancer.feature_names
```

Out[6]:

```
array(['mean radius', 'mean texture', 'mean perimeter', 'mean area',
       'mean smoothness', 'mean compactness', 'mean concavity',
       'mean concave points', 'mean symmetry', 'mean fractal dimension',
       'radius error', 'texture error', 'perimeter error', 'area error',
       'smoothness error', 'compactness error', 'concavity error',
       'concave points error', 'symmetry error',
       'fractal dimension error', 'worst radius', 'worst texture',
       'worst perimeter', 'worst area', 'worst smoothness',
       'worst compactness', 'worst concavity', 'worst concave points',
       'worst symmetry', 'worst fractal dimension'], dtype='<U23')
```

Class labels are:

In [7]:

```
cancer.target_names
```

Out[7]:

```
array(['malignant', 'benign'], dtype='<U9')
```

Create dataset for classification

In [9]:

```
X = cancer.data
Y = cancer.target
```

Number of samples are:

In [10]:

```
X.shape
```

Out[10]:

```
(569, 30)
```

In [11]:

```
print(type(cancer))
```

```
<class 'sklearn.utils.Bunch'>
```

Question 8a: Of all the SVM classifiers you explored in this hands-on exercise (i.e., linear SVM, SVM with a polynomial kernel and RBF kernel), which SVM results in a highest 10-fold cross-validation accuracy on this dataset? Explore the possible parameters for each SVM to determine the best performance for that SVM. For example, when studying linear SVM, explore a range of C values [0.001, 0.01, 0.1, 1]. Similarly for degree consider [1,2]. For gamma, consider [0.001, 0.01, 0.1, 1, 10, 100].

In [11]:

```
#Linear SVM
c=[0.001,0.01,0.1,1]
for x in c:
    svm_linear=SVC(C=x,kernel='linear')
    scores = cross_val_score(svm_linear, X, Y, cv=10, scoring='accuracy')
    print('Linear SVM Classifier with regularization constant '+ str(x) + ' has an accu
racy of \n' +str(scores*100) +' and its mean =  '+str(scores.mean()*100))
```

```
Linear SVM Classifier with regularization constant 0.001 has an accuracy
of
[ 91.37931034  91.37931034  94.73684211  94.73684211 100.
  98.24561404  92.98245614  89.28571429  92.85714286  94.64285714] and i
ts mean =  94.02460893613342
Linear SVM Classifier with regularization constant 0.01 has an accuracy
of
[ 96.55172414  91.37931034  94.73684211  94.73684211 100.
  96.49122807  92.98245614  89.28571429  96.42857143  94.64285714] and i
ts mean =  94.72355457609541
Linear SVM Classifier with regularization constant 0.1 has an accuracy o
f
[96.55172414 93.10344828 92.98245614 94.73684211 98.24561404 96.49122807
 92.98245614 91.07142857 96.42857143 94.64285714] and its mean =  94.723
66260478783
Linear SVM Classifier with regularization constant 1 has an accuracy of
[98.27586207 93.10344828 92.98245614 94.73684211 96.49122807 98.24561404
 92.98245614 94.64285714 96.42857143 96.42857143] and its mean =  95.431
79068360554
```

In [12]:

```
#poly kernel varying c and p]
poly=[1,2]
for x in c:
    for p in poly:
        svm_poly=SVC(C=x,kernel='poly',degree=p,gamma='auto')
        scores1 = cross_val_score(svm_poly, X, Y, cv=10, scoring='accuracy')
        print('Non Linear SVM Classifier with regularization constant '+ str(x) +' with
kernel of type poly with degree ' + str(p)  + ' has an accuracy of \n' +str(scores1*100
) +' and its mean = '+str(scores1.mean()*100))
```

```
Non Linear SVM Classifier with regularization constant 0.001 with kernel o
f type poly with degree 1 has an accuracy of
[91.37931034 89.65517241 91.22807018 94.73684211 94.73684211 89.47368421
 98.24561404 92.85714286 89.28571429 96.42857143] and its mean = 92.802696
39616282
Non Linear SVM Classifier with regularization constant 0.001 with kernel o
f type poly with degree 2 has an accuracy of
[98.27586207 91.37931034 91.22807018 92.98245614 96.49122807 98.24561404
 92.98245614 94.64285714 96.42857143 96.42857143] and its mean = 94.908499
69751964
Non Linear SVM Classifier with regularization constant 0.01 with kernel of
type poly with degree 1 has an accuracy of
[93.10344828 91.37931034 94.73684211 92.98245614 98.24561404 92.98245614
 96.49122807 91.07142857 92.85714286 96.42857143] and its mean = 94.027849
79690605
Non Linear SVM Classifier with regularization constant 0.01 with kernel of
type poly with degree 2 has an accuracy of
[ 98.27586207  91.37931034  91.22807018  96.49122807  96.49122807
  98.24561404  94.73684211  94.64285714 100.          96.42857143] and its
mean = 95.79195834413619
Non Linear SVM Classifier with regularization constant 0.1 with kernel of
type poly with degree 1 has an accuracy of
[ 96.55172414  91.37931034  94.73684211  94.73684211 100.
  94.73684211  92.98245614  91.07142857  96.42857143  94.64285714] and its
mean = 94.72668740817561
Non Linear SVM Classifier with regularization constant 0.1 with kernel of
type poly with degree 2 has an accuracy of
[98.27586207 91.37931034 91.22807018 96.49122807 96.49122807 98.24561404
 94.73684211 96.42857143 98.21428571 96.42857143] and its mean = 95.791958
34413619
Non Linear SVM Classifier with regularization constant 1 with kernel of ty
pe poly with degree 1 has an accuracy of
[96.55172414 91.37931034 94.73684211 94.73684211 98.24561404 96.49122807
 92.98245614 91.07142857 96.42857143 94.64285714] and its mean = 94.726687
40817561
Non Linear SVM Classifier with regularization constant 1 with kernel of ty
pe poly with degree 2 has an accuracy of
[ 98.27586207  91.37931034  92.98245614  96.49122807  96.49122807
  94.73684211  94.73684211  96.42857143 100.          96.42857143] and its
mean = 95.79509117621639
```

In [10]:

```
#RBF varying C and gamma
c=[0.001,0.01,0.1,1]
gamma=[0.001, 0.01, 0.1, 1, 10, 100]
for x in c:
    for g in gamma:
        svm_rbf=SVC(C=x,kernel='rbf',gamma=g)
        scores2 = cross_val_score(svm_rbf, X, Y, cv=10, scoring='accuracy')
        print('Non Linear SVM Classifier with regularization constant '+ str(x) +' and
kernel of type rbf with gamma ' + str(gamma)  + ' has an accuracy of \n' +str(scores2*1
00) +' and its mean = '+str(scores2.mean()*100))
```

```
Non Linear SVM Classifier with regularization constant 0.001 and  kernel o
f type rbf with gamma [0.001, 0.01, 0.1, 1, 10, 100] has an accuracy of
[62.06896552 62.06896552 63.15789474 63.15789474 63.15789474 63.15789474
 63.15789474 62.5        62.5        62.5       ] and its mean = 62.742740
47186933
Non Linear SVM Classifier with regularization constant 0.001 and  kernel o
f type rbf with gamma [0.001, 0.01, 0.1, 1, 10, 100] has an accuracy of
[62.06896552 62.06896552 63.15789474 63.15789474 63.15789474 63.15789474
 63.15789474 62.5        62.5        62.5       ] and its mean = 62.742740
47186933
Non Linear SVM Classifier with regularization constant 0.001 and  kernel o
f type rbf with gamma [0.001, 0.01, 0.1, 1, 10, 100] has an accuracy of
[62.06896552 62.06896552 63.15789474 63.15789474 63.15789474 63.15789474
 63.15789474 62.5        62.5        62.5       ] and its mean = 62.742740
47186933
Non Linear SVM Classifier with regularization constant 0.001 and  kernel o
f type rbf with gamma [0.001, 0.01, 0.1, 1, 10, 100] has an accuracy of
[62.06896552 62.06896552 63.15789474 63.15789474 63.15789474 63.15789474
 63.15789474 62.5        62.5        62.5       ] and its mean = 62.742740
47186933
Non Linear SVM Classifier with regularization constant 0.001 and  kernel o
f type rbf with gamma [0.001, 0.01, 0.1, 1, 10, 100] has an accuracy of
[62.06896552 62.06896552 63.15789474 63.15789474 63.15789474 63.15789474
 63.15789474 62.5        62.5        62.5       ] and its mean = 62.742740
47186933
Non Linear SVM Classifier with regularization constant 0.001 and  kernel o
f type rbf with gamma [0.001, 0.01, 0.1, 1, 10, 100] has an accuracy of
[62.06896552 62.06896552 63.15789474 63.15789474 63.15789474 63.15789474
 63.15789474 62.5        62.5        62.5       ] and its mean = 62.742740
47186933
Non Linear SVM Classifier with regularization constant 0.01 and  kernel of
type rbf with gamma [0.001, 0.01, 0.1, 1, 10, 100] has an accuracy of
[62.06896552 62.06896552 63.15789474 63.15789474 63.15789474 63.15789474
 63.15789474 62.5        62.5        62.5       ] and its mean = 62.742740
47186933
Non Linear SVM Classifier with regularization constant 0.01 and  kernel of
type rbf with gamma [0.001, 0.01, 0.1, 1, 10, 100] has an accuracy of
[62.06896552 62.06896552 63.15789474 63.15789474 63.15789474 63.15789474
 63.15789474 62.5        62.5        62.5       ] and its mean = 62.742740
47186933
Non Linear SVM Classifier with regularization constant 0.01 and  kernel of
type rbf with gamma [0.001, 0.01, 0.1, 1, 10, 100] has an accuracy of
[62.06896552 62.06896552 63.15789474 63.15789474 63.15789474 63.15789474
 63.15789474 62.5        62.5        62.5       ] and its mean = 62.742740
47186933
Non Linear SVM Classifier with regularization constant 0.01 and  kernel of
type rbf with gamma [0.001, 0.01, 0.1, 1, 10, 100] has an accuracy of
[62.06896552 62.06896552 63.15789474 63.15789474 63.15789474 63.15789474
 63.15789474 62.5        62.5        62.5       ] and its mean = 62.742740
47186933
Non Linear SVM Classifier with regularization constant 0.01 and  kernel of
type rbf with gamma [0.001, 0.01, 0.1, 1, 10, 100] has an accuracy of
[62.06896552 62.06896552 63.15789474 63.15789474 63.15789474 63.15789474
 63.15789474 62.5        62.5        62.5       ] and its mean = 62.742740
47186933
Non Linear SVM Classifier with regularization constant 0.01 and  kernel of
type rbf with gamma [0.001, 0.01, 0.1, 1, 10, 100] has an accuracy of
[62.06896552 62.06896552 63.15789474 63.15789474 63.15789474 63.15789474
 63.15789474 62.5        62.5        62.5       ] and its mean = 62.742740
47186933
Non Linear SVM Classifier with regularization constant 0.1 and  kernel of
```

```
type rbf with gamma [0.001, 0.01, 0.1, 1, 10, 100] has an accuracy of
[62.06896552 62.06896552 63.15789474 63.15789474 63.15789474 63.15789474
 63.15789474 62.5        62.5        62.5       ] and its mean = 62.742740
47186933
Non Linear SVM Classifier with regularization constant 0.1 and  kernel of
type rbf with gamma [0.001, 0.01, 0.1, 1, 10, 100] has an accuracy of
[62.06896552 62.06896552 63.15789474 63.15789474 63.15789474 63.15789474
 63.15789474 62.5        62.5        62.5       ] and its mean = 62.742740
47186933
Non Linear SVM Classifier with regularization constant 0.1 and  kernel of
type rbf with gamma [0.001, 0.01, 0.1, 1, 10, 100] has an accuracy of
[62.06896552 62.06896552 63.15789474 63.15789474 63.15789474 63.15789474
 63.15789474 62.5        62.5        62.5       ] and its mean = 62.742740
47186933
Non Linear SVM Classifier with regularization constant 0.1 and  kernel of
type rbf with gamma [0.001, 0.01, 0.1, 1, 10, 100] has an accuracy of
[62.06896552 62.06896552 63.15789474 63.15789474 63.15789474 63.15789474
 63.15789474 62.5        62.5        62.5       ] and its mean = 62.742740
47186933
Non Linear SVM Classifier with regularization constant 0.1 and  kernel of
type rbf with gamma [0.001, 0.01, 0.1, 1, 10, 100] has an accuracy of
[62.06896552 62.06896552 63.15789474 63.15789474 63.15789474 63.15789474
 63.15789474 62.5        62.5        62.5       ] and its mean = 62.742740
47186933
Non Linear SVM Classifier with regularization constant 0.1 and  kernel of
type rbf with gamma [0.001, 0.01, 0.1, 1, 10, 100] has an accuracy of
[62.06896552 62.06896552 63.15789474 63.15789474 63.15789474 63.15789474
 63.15789474 62.5        62.5        62.5       ] and its mean = 62.742740
47186933
Non Linear SVM Classifier with regularization constant 1 and  kernel of ty
pe rbf with gamma [0.001, 0.01, 0.1, 1, 10, 100] has an accuracy of
[94.82758621 94.82758621 91.22807018 92.98245614 92.98245614 91.22807018
 94.73684211 92.85714286 89.28571429 89.28571429] and its mean = 92.424163
85792065
Non Linear SVM Classifier with regularization constant 1 and  kernel of ty
pe rbf with gamma [0.001, 0.01, 0.1, 1, 10, 100] has an accuracy of
[62.06896552 63.79310345 64.9122807  64.9122807  63.15789474 63.15789474
 63.15789474 64.28571429 60.71428571 62.5       ] and its mean = 63.266031
45795523
Non Linear SVM Classifier with regularization constant 1 and  kernel of ty
pe rbf with gamma [0.001, 0.01, 0.1, 1, 10, 100] has an accuracy of
[62.06896552 62.06896552 63.15789474 63.15789474 63.15789474 63.15789474
 63.15789474 62.5        62.5        62.5       ] and its mean = 62.742740
47186933
Non Linear SVM Classifier with regularization constant 1 and  kernel of ty
pe rbf with gamma [0.001, 0.01, 0.1, 1, 10, 100] has an accuracy of
[62.06896552 62.06896552 63.15789474 63.15789474 63.15789474 63.15789474
 63.15789474 62.5        62.5        62.5       ] and its mean = 62.742740
47186933
Non Linear SVM Classifier with regularization constant 1 and  kernel of ty
pe rbf with gamma [0.001, 0.01, 0.1, 1, 10, 100] has an accuracy of
[62.06896552 62.06896552 63.15789474 63.15789474 63.15789474 63.15789474
 63.15789474 62.5        62.5        62.5       ] and its mean = 62.742740
47186933
Non Linear SVM Classifier with regularization constant 1 and  kernel of ty
pe rbf with gamma [0.001, 0.01, 0.1, 1, 10, 100] has an accuracy of
[62.06896552 62.06896552 63.15789474 63.15789474 63.15789474 63.15789474
 63.15789474 62.5        62.5        62.5       ] and its mean = 62.742740
47186933
```

Answer: Non Linear SVM Classifier with regularization constant c=1 with kernel of type poly with degree 2 has better accuracy when compared with other models and hyperparameters.

Question 8b: Similar to **Question 8a** explore decision trees with different max_depth to determine which values returns the best classifier.

In [12]:

```
max_depth=[2,4,6,8,10,20,30,40]
for x in max_depth:
    dt = DecisionTreeClassifier(max_depth=x)
    scores_dt = cross_val_score(dt, X, Y, cv=10, scoring='accuracy')
    print('Decision Tree with depth ' + str(x)+' on the cancer dataset ' +str(scores_dt
*100) +' and its mean = ' +str(scores_dt.mean()*100))
```

```
Decision Tree with depth 2 on the cancer dataset [91.37931034 86.20689655
89.47368421 91.22807018 94.73684211 91.22807018
 92.98245614 92.85714286 94.64285714 96.42857143] and its mean = 92.116390
11321407
Decision Tree with depth 4 on the cancer dataset [ 94.82758621  84.4827586
2  92.98245614  87.71929825  96.49122807
  91.22807018  89.47368421  92.85714286  91.07142857 100.        ] and its
mean = 92.1133653098263
Decision Tree with depth 6 on the cancer dataset [94.82758621 87.93103448
92.98245614 87.71929825 96.49122807 89.47368421
 87.71929825 94.64285714 91.07142857 96.42857143] and its mean = 91.928744
27447929
Decision Tree with depth 8 on the cancer dataset [94.82758621 86.20689655
92.98245614 89.47368421 98.24561404 89.47368421
 87.71929825 94.64285714 91.07142857 89.28571429] and its mean = 91.392921
9600726
Decision Tree with depth 10 on the cancer dataset [89.65517241 87.93103448
91.22807018 85.96491228 92.98245614 87.71929825
 87.71929825 94.64285714 92.85714286 98.21428571] and its mean = 90.891452
76985568
Decision Tree with depth 20 on the cancer dataset [91.37931034 89.65517241
91.22807018 89.47368421 94.73684211 89.47368421
 91.22807018 94.64285714 92.85714286 85.71428571] and its mean = 91.038911
93500993
Decision Tree with depth 30 on the cancer dataset [94.82758621 86.20689655
91.22807018 87.71929825 96.49122807 91.22807018
 87.71929825 94.64285714 91.07142857 94.64285714] and its mean = 91.577759
05280442
Decision Tree with depth 40 on the cancer dataset [94.82758621 87.93103448
91.22807018 89.47368421 94.73684211 91.22807018
 89.47368421 94.64285714 91.07142857 91.07142857] and its mean = 91.568468
58525626
```

Answer: when depth=2 the decision tree classifier, classifies the data with better accuracy.

Question 8c: Imagine a scenario where you are working at a cancer center as a data scientist tasked with identifying the characteristics that distinguish malignant tumors from benign tumors. Based on your knowledge of classification techniques which approach would you use and why?

Answer: I will choose SVM classification technique for this problem because from the choice of classification techniques will remove KNN as it is 30D data and when we choose KNN we have to face the curse of dimensionality .Coming to naive bayes classifier I didn't choose because in naive bayes we assume attributes to be independent to each other and in this data mean area and mean perimeter values depend on the radius attribute so I have n't choosen that. so I need to choose among decision tree and SVM, as decision trees are very sensitive to small variations in the training data and when calculated the accuracy between SVM and DT, we are getting better accuracy with SVM.