Hands-on Exercise CLASS Module

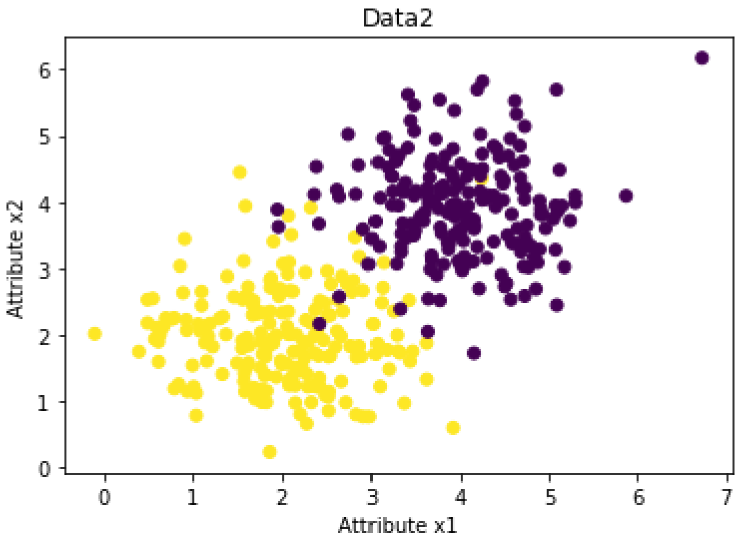
In [1]:
```
!pip install --user mlxtend
```

```
Requirement already satisfied: mlxtend in /users/PES0801/shravreddy/.local/li
b/python3.6/site-packages
Requirement already satisfied: scikit-learn>=0.20.3 in /users/PES0801/shravre
ddy/.local/lib/python3.6/site-packages (from mlxtend)
Requirement already satisfied: numpy>=1.16.2 in /users/PES0801/shravreddy/.lo
cal/lib/python3.6/site-packages (from mlxtend)
Requirement already satisfied: pandas>=0.24.2 in /users/PES0801/shravreddy/.l
ocal/lib/python3.6/site-packages (from mlxtend)
Requirement already satisfied: matplotlib>=3.0.0 in /users/PES0801/shravredd
y/.local/lib/python3.6/site-packages (from mlxtend)
Requirement already satisfied: setuptools in /usr/local/anaconda5/lib/python
3.6/site-packages (from mlxtend)
Requirement already satisfied: joblib>=0.13.2 in /users/PES0801/shravreddy/.l
ocal/lib/python3.6/site-packages (from mlxtend)
Requirement already satisfied: scipy>=1.2.1 in /users/PES0801/shravreddy/.loc
al/lib/python3.6/site-packages (from mlxtend)
Requirement already satisfied: python-dateutil>=2.6.1 in /usr/local/anaconda
5/lib/python3.6/site-packages (from pandas>=0.24.2->mlxtend)
Requirement already satisfied: pytz>=2017.2 in /usr/local/anaconda5/lib/pytho
n3.6/site-packages (from pandas>=0.24.2->mlxtend)
Requirement already satisfied: kiwisolver>=1.0.1 in /users/PES0801/shravredd
y/.local/lib/python3.6/site-packages (from matplotlib>=3.0.0->mlxtend)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in /u
sr/local/anaconda5/lib/python3.6/site-packages (from matplotlib>=3.0.0->mlxte
nd)
Requirement already satisfied: cycler>=0.10 in /usr/local/anaconda5/lib/pytho
n3.6/site-packages (from matplotlib>=3.0.0->mlxtend)
Requirement already satisfied: six>=1.5 in /usr/local/anaconda5/lib/python3.
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.
```

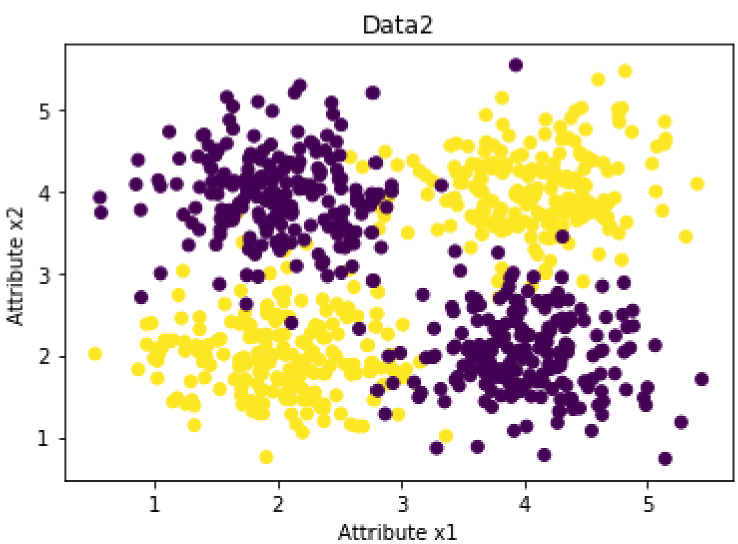
In [4]:
```
import numpy as np

#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 [5]:
```
# 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],Data
4_X[:,1]) - 100 < 0 ] = 0
```

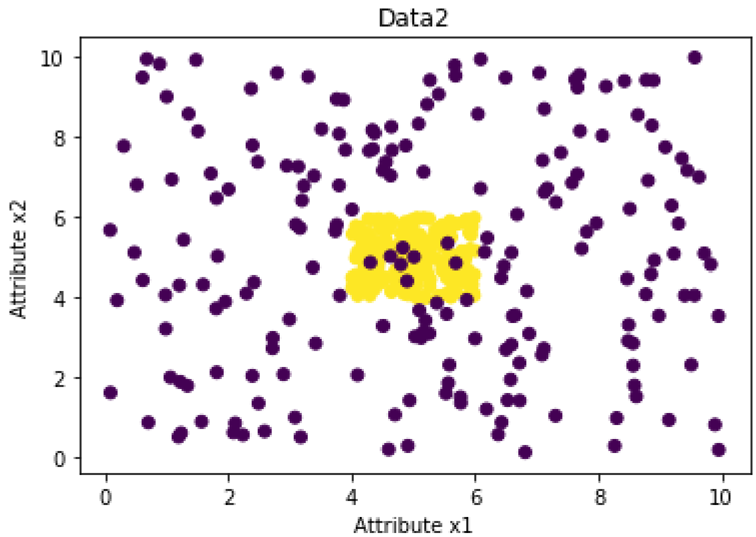
In [6]:
```
plt.scatter(Data1_X[:,0],Data1_X[:,1], c= Data1_Y)
plt.xlabel('Attribute x1')
plt.ylabel('Attribute x2')
plt.title('Data2')
plt.show()
```



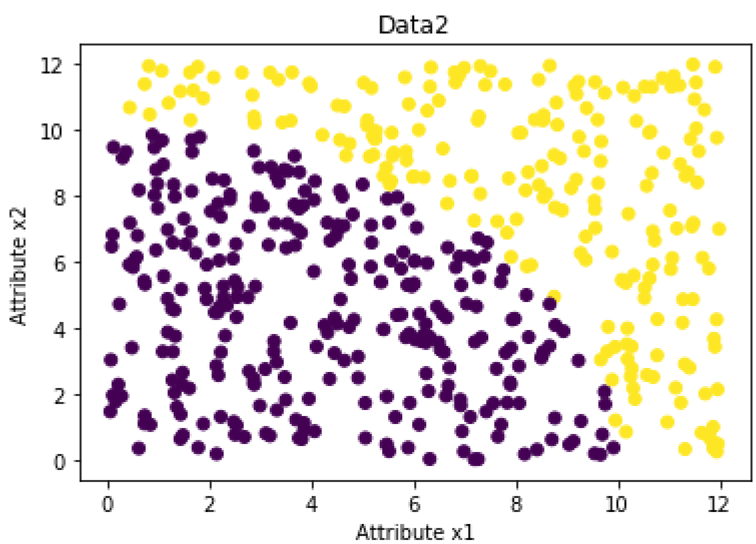
In [5]:
```
plt.scatter(Data2_X[:,0],Data2_X[:,1], c= Data2_Y)
plt.xlabel('Attribute x1')
plt.ylabel('Attribute x2')
plt.title('Data2')
plt.show()
```



In [6]:
```
plt.scatter(Data3_X[:,0],Data3_X[:,1], c= Data3_Y)
plt.xlabel('Attribute x1')
plt.ylabel('Attribute x2')
plt.title('Data2')
plt.show()
```



In [7]:
```
plt.scatter(Data4_X[:,0],Data4_X[:,1], c= Data4_Y)
plt.xlabel('Attribute x1')
plt.ylabel('Attribute x2')
plt.title('Data2')
plt.show()
```



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 [31]:
```
dt3_2 = DecisionTreeClassifier(max_depth=2)
dt_scores3_2 = cross_val_score(dt3_2, Data3_X, Data3_Y, cv=10, scoring='accura
cy')
[dt_scores3_2,dt_scores3_2.mean()]
```

Out[31]:
```
[array([0.8  , 0.9  , 0.85 , 0.875, 0.925, 0.875, 0.9  , 0.85 , 0.85 ,
        0.9  ]), 0.8724999999999999]
```

In [32]:
```
dt3_4 = DecisionTreeClassifier(max_depth=4)
dt_scores3_4 = cross_val_score(dt3_4, Data3_X, Data3_Y, cv=10, scoring='accura
cy')
[dt_scores3_4,dt_scores3_4.mean()]
```

Out[32]:
```
[array([0.95 , 0.975, 0.975, 0.975, 1.   , 1.   , 0.975, 0.925, 0.975,
        0.975]), 0.9724999999999999]
```

In [33]:
```
dt3_6 = DecisionTreeClassifier(max_depth=6)
dt_scores3_6 = cross_val_score(dt3_6, Data3_X, Data3_Y, cv=10, scoring='accura
cy')
[dt_scores3_6,dt_scores3_6.mean()]
```

Out[33]:
```
[array([0.95 , 0.975, 0.975, 0.975, 1.   , 1.   , 0.95 , 0.925, 0.975,
        0.975]), 0.97]
```

In [34]:
```
dt3_8 = DecisionTreeClassifier(max_depth=8)
dt_scores3_8 = cross_val_score(dt3_8, Data3_X, Data3_Y, cv=10, scoring='accura
cy')
[dt_scores3_8,dt_scores3_8.mean()]
```

Out[34]:
```
[array([0.95 , 0.975, 0.95 , 0.95 , 1.   , 0.975, 0.9  , 0.875, 0.975,
        0.975]), 0.9524999999999999]
```

In [35]:
```
dt3_10 = DecisionTreeClassifier(max_depth=10)
dt_scores3_10 = cross_val_score(dt3_10, Data3_X, Data3_Y, cv=10, scoring='accu
racy')
[dt_scores3_10,dt_scores3_10.mean()]
```

Out[35]:
```
[array([0.925, 0.95 , 0.95 , 0.95 , 1.   , 0.975, 0.9  , 0.875, 0.95 ,
        0.925]), 0.9400000000000001]
```

In [36]:
```
dt3_50 = DecisionTreeClassifier(max_depth=50)
dt_scores3_50 = cross_val_score(dt3_50, Data3_X, Data3_Y, cv=10, scoring='accu
racy')
[dt_scores3_50,dt_scores3_50.mean()]
```

Out[36]:
```
[array([0.925, 0.95 , 0.925, 0.95 , 1.   , 0.975, 0.9  , 0.9  , 0.95 ,
        0.925]), 0.9400000000000001]
```

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

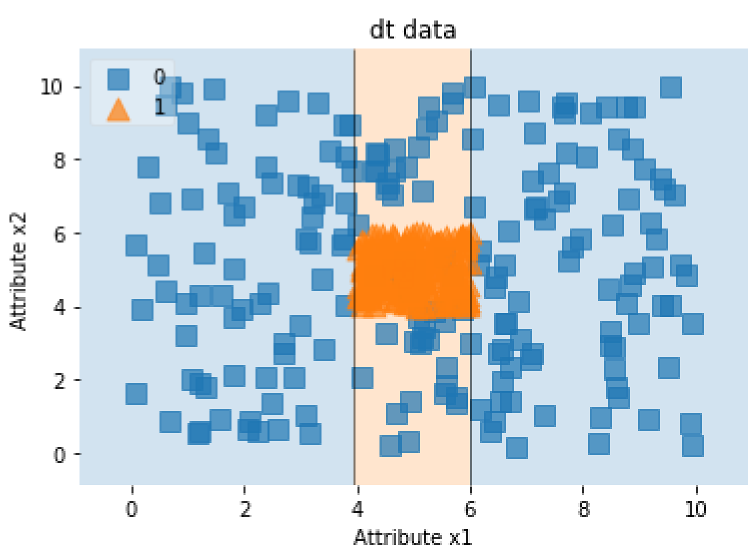
Answer: max_depth=2 has lowest accuracy. The phenomenone is called Underfitting.

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?

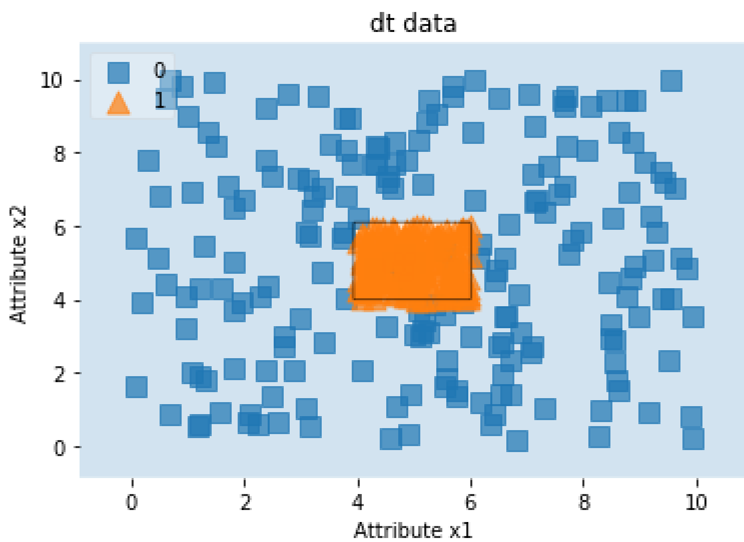
Answer: For max depth =50, accuracy is 0.94. Difference between highest and this accracy is 0.032. the phenomenone is Overfitting.

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

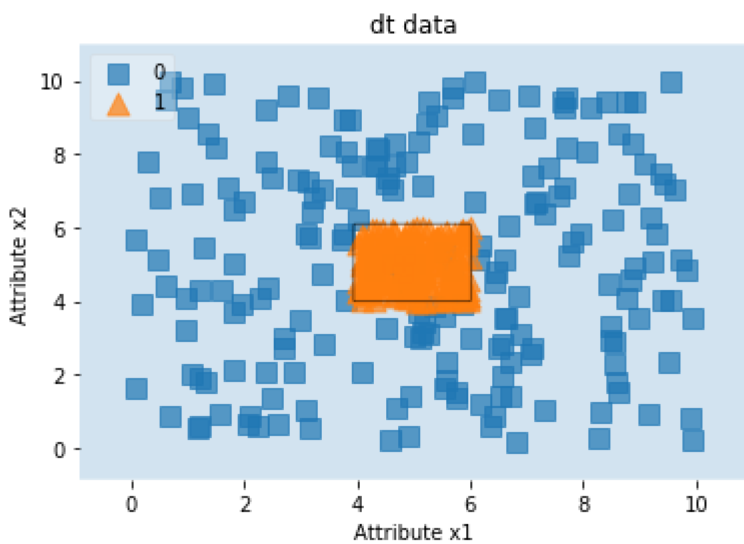
In [37]:
```
dt3_2.fit( Data3_X, Data3_Y)

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}

plot_decision_regions(X=Data3_X, y=Data3_Y, clf=dt3_2, 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('dt data')
plt.show()
```



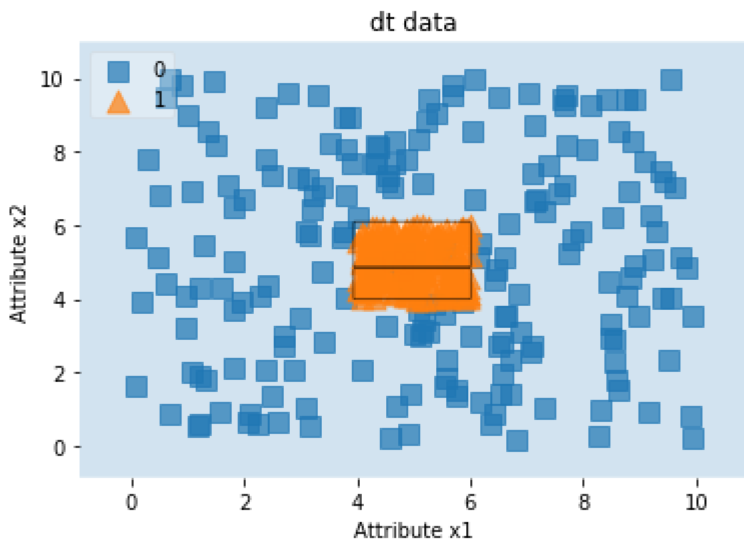
In [38]:
```
dt3_4.fit( Data3_X, Data3_Y)

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}

plot_decision_regions(X=Data3_X, y=Data3_Y, clf=dt3_4, 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('dt data')
plt.show()
```



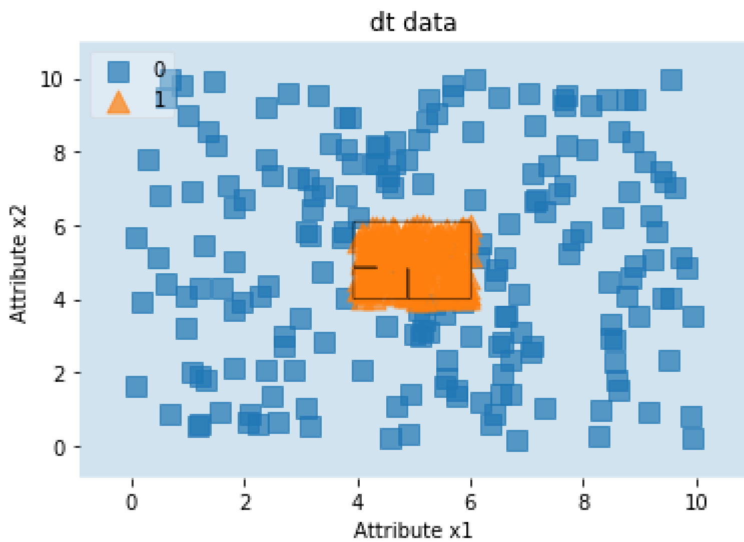
In [39]:
```
dt3_6.fit( Data3_X, Data3_Y)

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}

plot_decision_regions(X=Data3_X, y=Data3_Y, clf=dt3_6, 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('dt data')
plt.show()
```



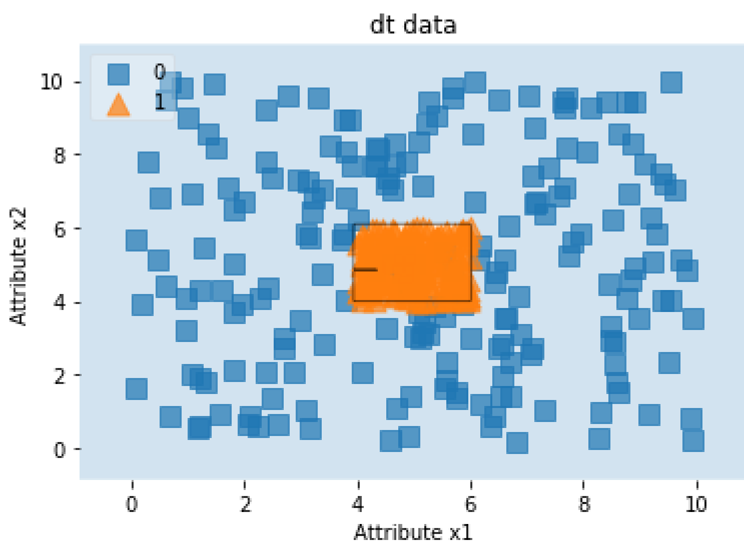
In [40]:
```
dt3_8.fit( Data3_X, Data3_Y)

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}

plot_decision_regions(X=Data3_X, y=Data3_Y, clf=dt3_8, 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('dt data')
plt.show()
```



In [41]:
```
dt3_10.fit( Data3_X, Data3_Y)

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}

plot_decision_regions(X=Data3_X, y=Data3_Y, clf=dt3_10, 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('dt data')
plt.show()
```



In [42]:
```
dt3_50.fit( Data3_X, Data3_Y)

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}

plot_decision_regions(X=Data3_X, y=Data3_Y, clf=dt3_50, 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('dt data')
plt.show()
```



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

Answer: For max depth=4 it is better suited fo the this data as it is able to divide the the two classes

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 [43]:
```
knn2_1 = KNeighborsClassifier(n_neighbors=1)
knn_scores2_1 = cross_val_score(knn2_1, Data2_X, Data2_Y, cv=10, scoring='accu
racy')
[knn_scores2_1,knn_scores2_1.mean()]
```

Out[43]:
```
[array([0.925 , 0.8875, 0.925 , 0.8875, 0.925 , 0.9375, 0.8875, 0.925 ,
        0.9375, 0.8875]), 0.9125]
```

In [50]:
```
knn2_5 = KNeighborsClassifier(n_neighbors=5)
knn_scores2_5 = cross_val_score(knn2_5, Data2_X, Data2_Y, cv=10, scoring='accu
racy')
[knn_scores2_5,knn_scores2_5.mean()]
```

Out[50]:
```
[array([0.9875, 0.9125, 0.925 , 0.9125, 0.95  , 0.95  , 0.8625, 0.95  ,
        0.9625, 0.9375]), 0.9349999999999999]
```

In [45]:
```
knn2_10 = KNeighborsClassifier(n_neighbors=10)
knn_scores2_10 = cross_val_score(knn2_10, Data2_X, Data2_Y, cv=10, scoring='ac
curacy')
[knn_scores2_10,knn_scores2_10.mean()]
```

Out[45]:
```
[array([0.9875, 0.9   , 0.95  , 0.925 , 0.9625, 0.95  , 0.8625, 0.9375,
        0.9625, 0.9625]), 0.9400000000000001]
```

In [52]:
```
knn2_50 = KNeighborsClassifier(n_neighbors=50)
knn_scores2_50 = cross_val_score(knn2_50, Data2_X, Data2_Y, cv=10, scoring='ac
curacy')
[knn_scores2_50,knn_scores2_50.mean()]
```

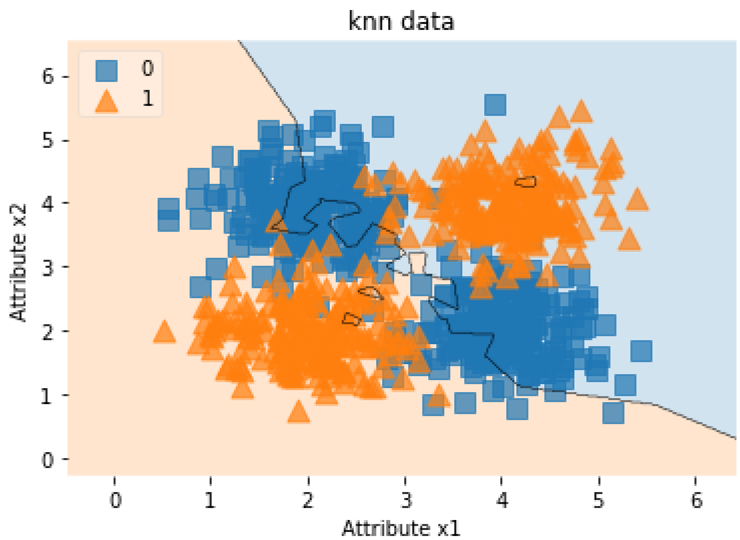
Out[52]:
```
[array([0.9875, 0.9   , 0.9625, 0.9125, 0.9625, 0.9375, 0.8875, 0.9375,
        0.9625, 0.9625]), 0.9412499999999999]
```

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

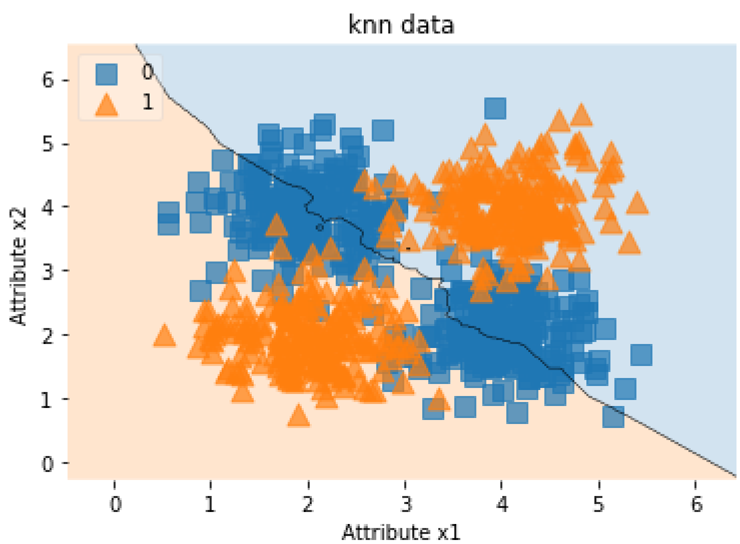
Answer: the lowest value of n_neighbours is observed for n=1. The phenomemone is underfitting.

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

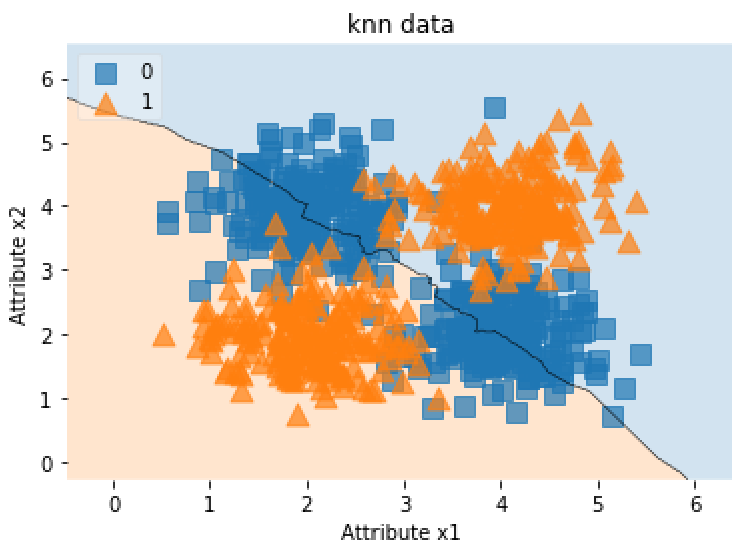
In [47]:
```
knn2_1.fit( Data1_X, Data1_Y)

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}

plot_decision_regions(X=Data2_X, y=Data2_Y, clf=knn2_1, 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('knn data')
plt.show()
```



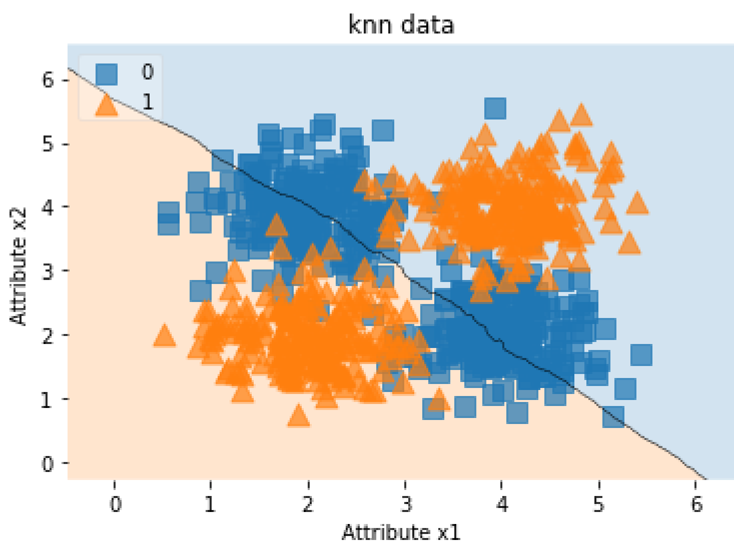
In [51]:
```
knn2_5.fit( Data1_X, Data1_Y)

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}

plot_decision_regions(X=Data2_X, y=Data2_Y, clf=knn2_5, 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('knn data')
plt.show()
```



In [49]:
```
knn2_10.fit( Data1_X, Data1_Y)

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}

plot_decision_regions(X=Data2_X, y=Data2_Y, clf=knn2_10, 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('knn data')
plt.show()
```



In [53]:
```
knn2_50.fit( Data1_X, Data1_Y)

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}

plot_decision_regions(X=Data2_X, y=Data2_Y, clf=knn2_50, 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('knn data')
plt.show()
```



In [ ]:
```
knn2_1.fit( Data1_X, Data1_Y)

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}

plot_decision_regions(X=Data2_X, y=Data2_Y, clf=knn2_1, 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('knn data')
plt.show()
```

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: The decision boundary became less curver and more to a straight line.

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 [16]:
```
nb1 = GaussianNB()
nb_scores1 = cross_val_score(nb1, Data1_X, Data1_Y, cv=10, scoring='accuracy')
[nb_scores1, nb_scores1.mean()]
```

Out[16]:
```
[array([0.975, 1.   , 1.   , 0.925, 0.95 , 0.975, 0.975, 0.9  , 0.975,
        1.   ]), 0.9675]
```

In [17]:
```
nb2 = GaussianNB()
nb_scores2 = cross_val_score(nb2, Data2_X, Data2_Y, cv=10, scoring='accuracy')
[nb_scores2, nb_scores2.mean()]
```

Out[17]:
```
[array([0.075 , 0.0625, 0.0125, 0.0875, 0.0875, 0.025 , 0.05  , 0.05  ,
        0.0125, 0.0375]), 0.049999999999999996]
```

In [18]:
```
nb3 = GaussianNB()
nb_scores3 = cross_val_score(nb3, Data3_X, Data3_Y, cv=10, scoring='accuracy')
[nb_scores3, nb_scores3.mean()]
```

Out[18]:
```
[array([1.   , 0.95 , 0.975, 0.975, 0.975, 0.975, 0.925, 0.9  , 0.975,
        0.95 ]), 0.96]
```

In [19]:
```
nb4 = GaussianNB()
nb_scores4 = cross_val_score(nb4, Data4_X, Data4_Y, cv=10, scoring='accuracy')
[nb_scores4, nb_scores4.mean()]
```

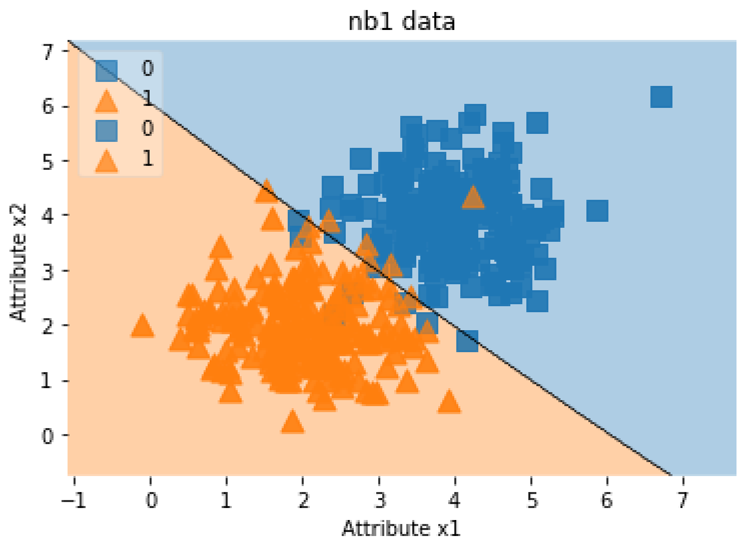
Out[19]:
```
[array([0.90196078, 1.        , 0.98      , 0.98      , 0.98      ,
        0.96      , 0.94      , 0.96      , 0.97959184, 0.95918367]),
 0.9640736294517807]
```

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

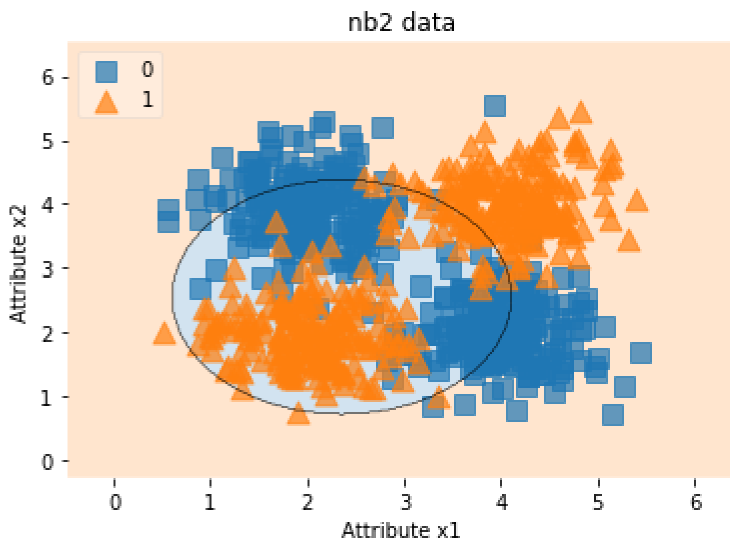
Answer: The Naive Bayes performed poorly on the dataset 2 because the points cannot be separated using this classifier effectively. As the points are scattered in such a way that those points of the same classes are concentrated diagonally opposite to each other making it difficult to classify them using Naive Bayes Classifier.

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

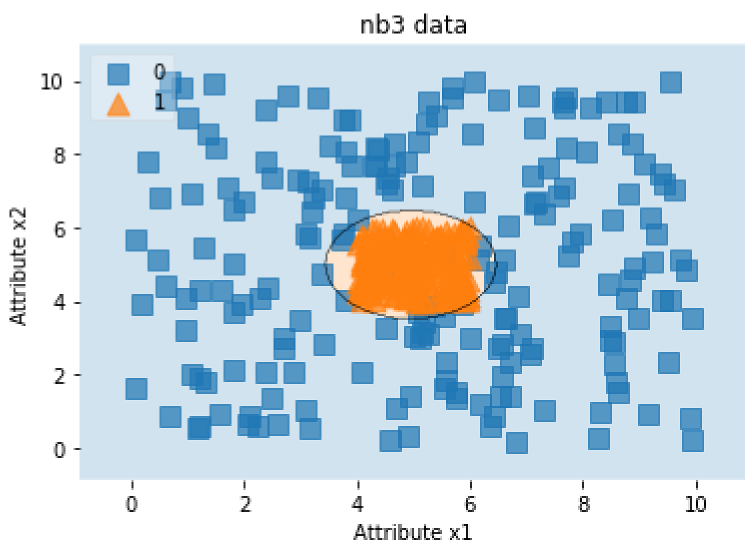
In [26]:
```
nb1.fit( Data1_X, Data1_Y)

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}

plot_decision_regions(X=Data1_X, y=Data1_Y, clf=nb1, 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('nb1 data')
plt.show()
```



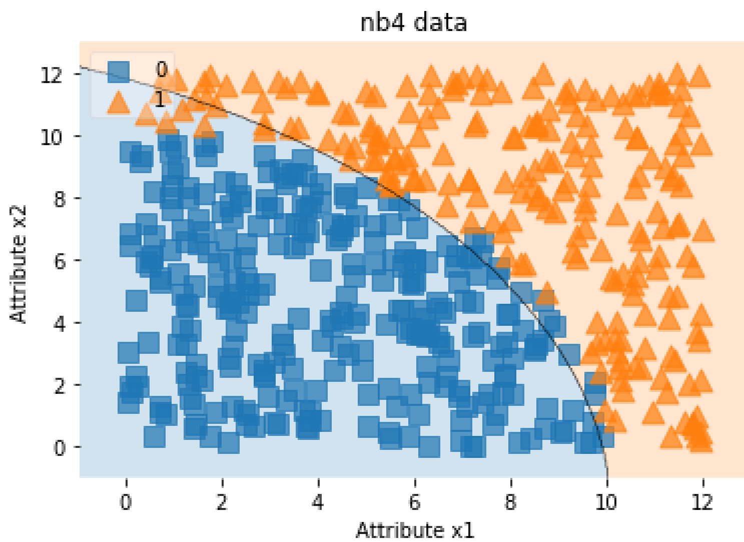
In [27]:
```
nb2.fit( Data2_X, Data2_Y)

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}

plot_decision_regions(X=Data2_X, y=Data2_Y, clf=nb2, 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('nb2 data')
plt.show()
```



In [28]:
```
nb3.fit( Data3_X, Data3_Y)

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}

plot_decision_regions(X=Data3_X, y=Data3_Y, clf=nb3, 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('nb3 data')
plt.show()
```



In [30]:
```
nb4.fit( Data4_X, Data4_Y)

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}

plot_decision_regions(X=Data4_X, y=Data4_Y, clf=nb4, 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('nb4 data')
plt.show()
```



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

Answer: The decision boundary of dataset 1 is a straight line as the points of same class are scattered in the same half. So a single line can separate the points of two classes. The decision boundary of dataset 2 is an ellipse. Since the points are concentrated in four circles and points of same classes are in such a way that they cannot be separated by single line. The decision boundary of dataset 3 is an ellipse. The points of one class are concentrated in the centre and the points of other class are situated around this class of points. So ellipse is an ideal separator of these classes. The decision boundary of dataset 4 is a parabola since the points of different classes can be separated by a curve.

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

Answer: In the datset 3, the decision boundary is unable to separate the classes effictively owing to the scattering of points in four circles with the points of same classes located in oppsite sides.

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.

Answer: Linear SVM performs well in datset 1 and datset 4 since most of the points can be separated by a line wheras in dataset 2 and 3 it is not possible to separate the points according to their classes. So the decrasing accuracies of the dataset are: Dataset1> Dataset4>Dataset 3> Dataset 2

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 [8]:
```
svm_linear1 = SVC(C=0.5, kernel='linear')
svm_linear_scores1 = cross_val_score(svm_linear1, Data1_X, Data1_Y, cv=10, sco
ring='accuracy')
[svm_linear_scores1, svm_linear_scores1.mean()]
```

Out[8]:
```
[array([0.975, 1.   , 1.   , 0.95 , 0.95 , 0.95 , 0.975, 0.9  , 0.975,
        1.   ]), 0.9674999999999999]
```

In [9]:
```
svm_linear2 = SVC(C=0.5, kernel='linear')
svm_linear_scores2 = cross_val_score(svm_linear2, Data2_X, Data2_Y, cv=10, sco
ring='accuracy')
[svm_linear_scores2, svm_linear_scores2.mean()]
```

Out[9]:
```
[array([0.125 , 0.1375, 0.0125, 0.0875, 0.2   , 0.2375, 0.1   , 0.15  ,
        0.1875, 0.175 ]), 0.14125000000000001]
```

In [10]:
```
svm_linear3 = SVC(C=0.5, kernel='linear')
svm_linear_scores3 = cross_val_score(svm_linear3, Data3_X, Data3_Y, cv=10, sco
ring='accuracy')
[svm_linear_scores3, svm_linear_scores3.mean()]
```

Out[10]:
```
[array([0.625, 0.625, 0.65 , 0.6  , 0.65 , 0.7  , 0.65 , 0.675, 0.625,
        0.625]), 0.6425000000000001]
```

In [11]:
```
svm_linear4 = SVC(C=0.5, kernel='linear')
svm_linear_scores4 = cross_val_score(svm_linear4, Data4_X, Data4_Y, cv=10, sco
ring='accuracy')
[svm_linear_scores4, svm_linear_scores4.mean()]
```

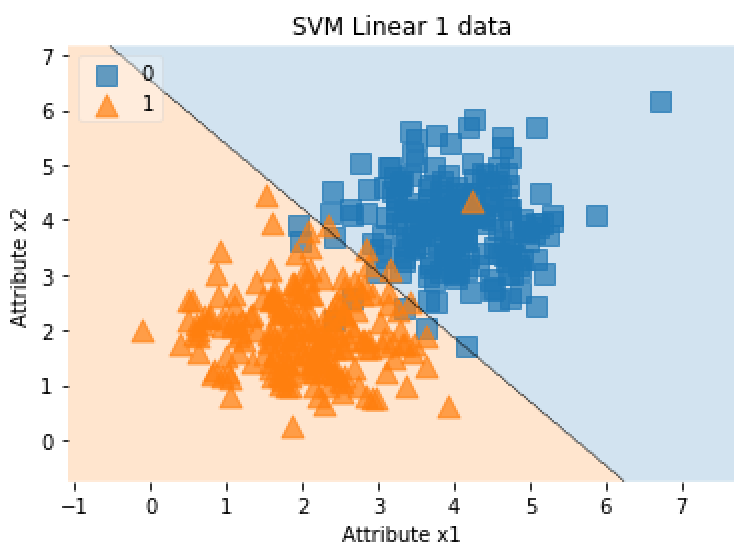
Out[11]:
```
[array([0.94117647, 0.90196078, 0.92      , 0.92      , 0.98      ,
        0.92      , 0.94      , 0.92      , 0.95918367, 0.85714286]),
 0.9259463785514207]
```

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

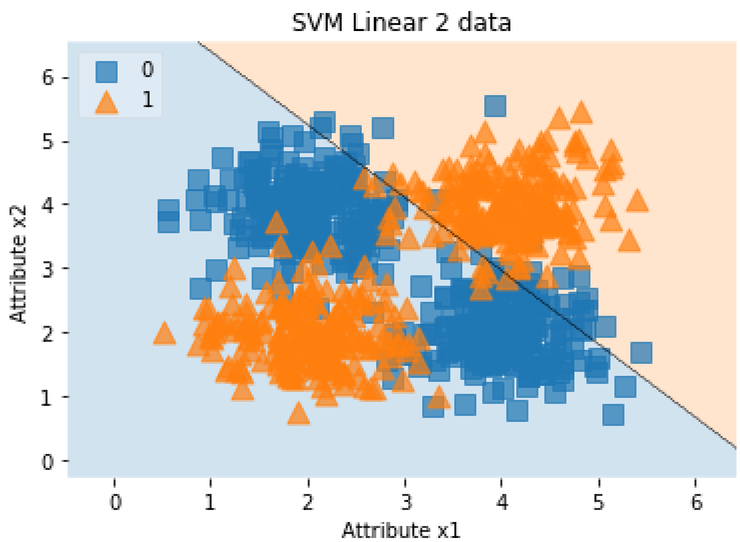
Answer: Dataset1 > Dataset4 > Dataset3 > Dataset2

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

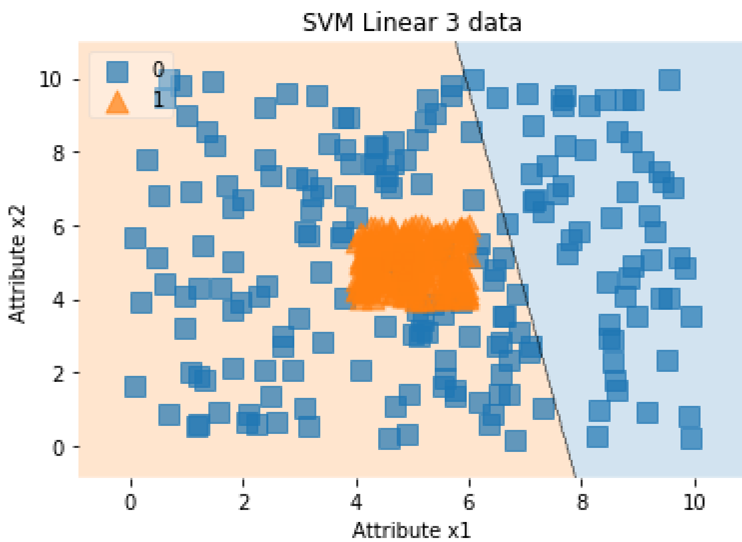
In [12]:
```
svm_linear1.fit( Data1_X, Data1_Y)

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}

plot_decision_regions(X=Data1_X, y=Data1_Y, clf=svm_linear1, 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('SVM Linear 1 data')
plt.show()
```



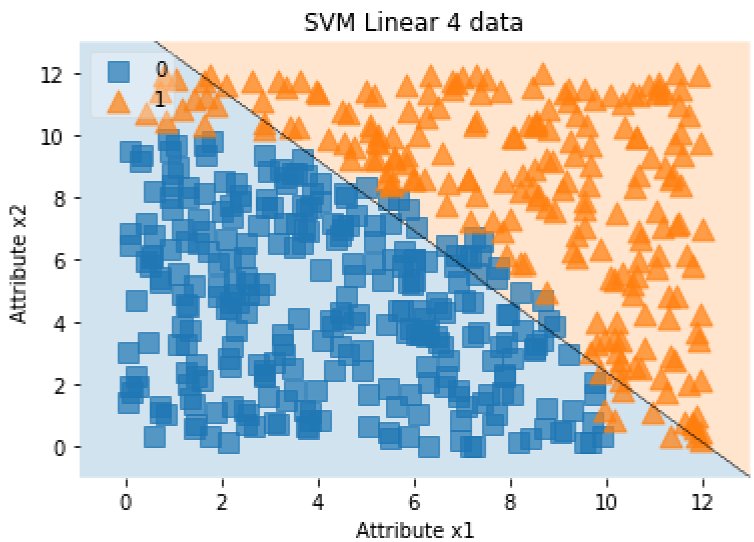
In [13]:
```
svm_linear2.fit( Data2_X, Data2_Y)

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}

plot_decision_regions(X=Data2_X, y=Data2_Y, clf=svm_linear2, 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('SVM Linear 2 data')
plt.show()
```



In [14]:
```
svm_linear3.fit( Data3_X, Data3_Y)

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}

plot_decision_regions(X=Data3_X, y=Data3_Y, clf=svm_linear3, 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('SVM Linear 3 data')
plt.show()
```



In [15]:
```
svm_linear4.fit( Data4_X, Data4_Y)

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}

plot_decision_regions(X=Data4_X, y=Data4_Y, clf=svm_linear4, 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('SVM Linear 4 data')
plt.show()
```



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

Answer: Since the two classes in datset are easily separable by a line , it has highest accuracy. Also the datsaet 2 resullts in a very bad decision boundary because of the presence of the points of the same classes.

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 [42]:
```
svm_poly2_1 = SVC(C=0.5, kernel='poly',degree=1, gamma = 'auto')
svm_poly_scores2_1 = cross_val_score(svm_poly2_1, Data2_X, Data2_Y, cv=10, sco
ring='accuracy')
[svm_poly_scores2_1, svm_poly_scores2_1.mean()]
```

Out[42]:
```
[array([0.1375, 0.125 , 0.0125, 0.0875, 0.175 , 0.1875, 0.1   , 0.1625,
        0.1875, 0.1625]), 0.13375]
```

In [43]:
```
svm_poly2_2 = SVC(C=0.5, kernel='poly',degree=2, gamma = 'auto')
svm_poly_scores2_2 = cross_val_score(svm_poly2_2, Data2_X, Data2_Y, cv=10, sco
ring='accuracy')
[svm_poly_scores2_2, svm_poly_scores2_2.mean()]
```

Out[43]:
```
[array([0.8125, 0.8375, 0.8875, 0.8375, 0.8875, 0.8875, 0.8625, 0.8875,
        0.9125, 0.8375]), 0.865]
```

In [44]:
```
svm_poly2_3 = SVC(C=0.5, kernel='poly',degree=3, gamma = 'auto')
svm_poly_scores2_3 = cross_val_score(svm_poly2_3, Data2_X, Data2_Y, cv=10, sco
ring='accuracy')
[svm_poly_scores2_3, svm_poly_scores2_3.mean()]
```

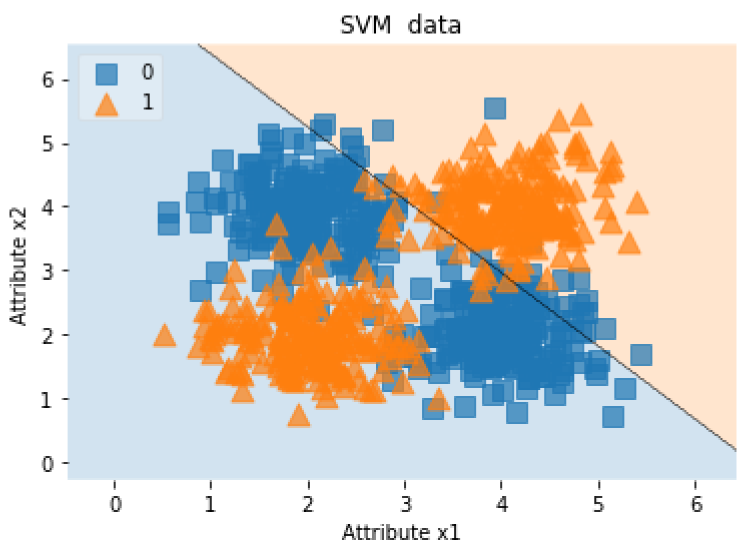
Out[44]:
```
[array([0.825 , 0.875 , 0.8875, 0.8625, 0.925 , 0.9   , 0.8625, 0.8875,
        0.8875, 0.85  ]), 0.8762500000000001]
```

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

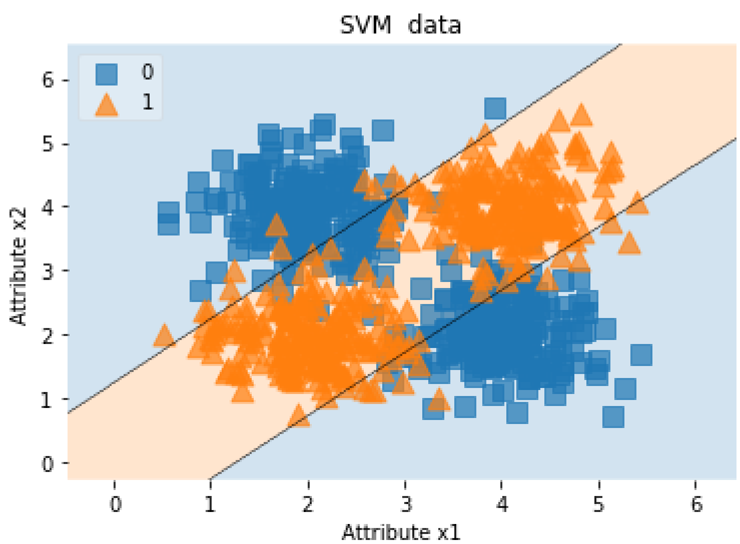
Answer: degree 3> degree 2> degree 1

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

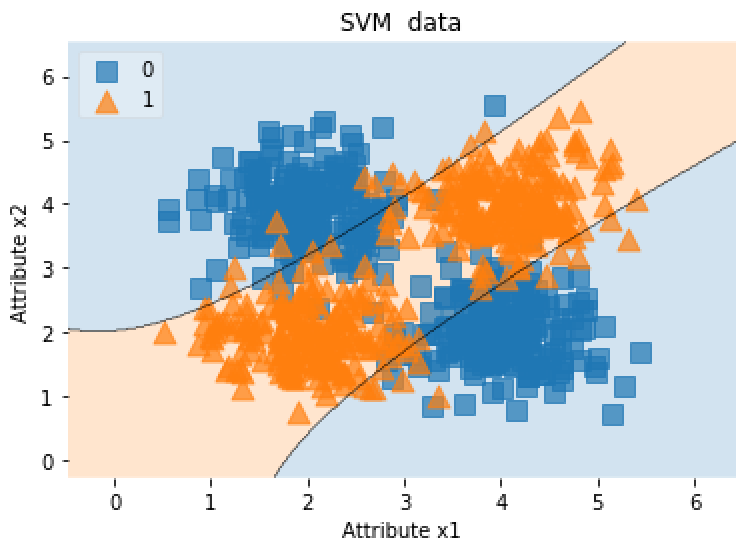
In [20]:
```
svm_poly2_1.fit( Data2_X, Data2_Y)

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}

plot_decision_regions(X=Data2_X, y=Data2_Y, clf=svm_poly2_1, 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('SVM  data')
plt.show()
```



In [21]:
```
svm_poly2_2.fit( Data2_X, Data2_Y)

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}

plot_decision_regions(X=Data2_X, y=Data2_Y, clf=svm_poly2_2, 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('SVM  data')
plt.show()
```



In [22]:
```
svm_poly2_3.fit( Data2_X, Data2_Y)

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}

plot_decision_regions(X=Data2_X, y=Data2_Y, clf=svm_poly2_3, 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('SVM  data')
plt.show()
```



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

Answer:

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 [23]:
```
svm_rbf2_001 = SVC(C = 0.5, kernel='rbf', gamma=0.01)
svm_rbf_scores2_001 = cross_val_score(svm_rbf2_001, Data2_X, Data2_Y, cv=10, s
coring='accuracy')
[svm_rbf_scores2_001, svm_rbf_scores2_001.mean()]
```

Out[23]:
```
[array([0.375 , 0.3125, 0.0875, 0.25  , 0.4375, 0.3375, 0.3   , 0.3   ,
        0.275 , 0.3375]), 0.30124999999999996]
```

In [24]:
```
svm_rbf2_01 = SVC(C = 0.5, kernel='rbf', gamma=0.1)
svm_rbf_scores2_01 = cross_val_score(svm_rbf2_01, Data2_X, Data2_Y, cv=10, sco
ring='accuracy')
[svm_rbf_scores2_01, svm_rbf_scores2_01.mean()]
```

Out[24]:
```
[array([0.975 , 0.9   , 0.9375, 0.9   , 0.9625, 0.9375, 0.8875, 0.9375,
        0.9625, 0.9625]), 0.93625]
```

In [25]:
```
svm_rbf2_1 = SVC(C = 0.5, kernel='rbf', gamma=1)
svm_rbf_scores2_1 = cross_val_score(svm_rbf2_1, Data2_X, Data2_Y, cv=10, scori
ng='accuracy')
[svm_rbf_scores2_1, svm_rbf_scores2_1.mean()]
```

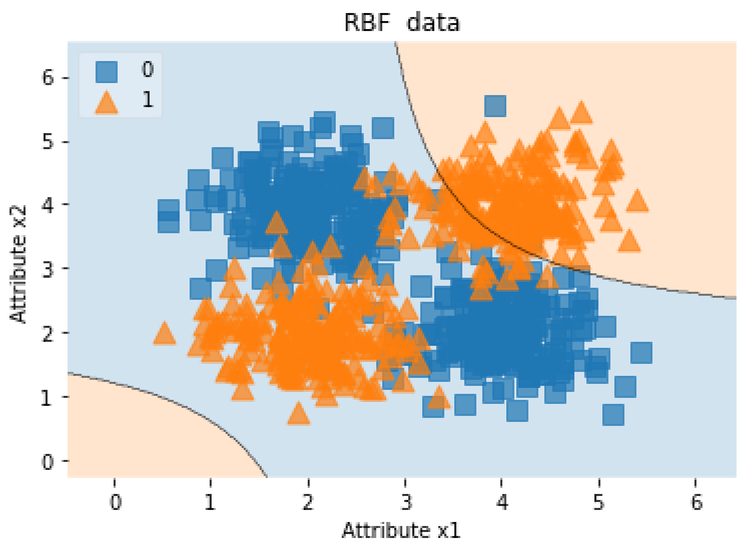
Out[25]:
```
[array([0.9875, 0.9125, 0.95  , 0.925 , 0.9625, 0.9375, 0.875 , 0.9375,
        0.9625, 0.95  ]), 0.9399999999999998]
```

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

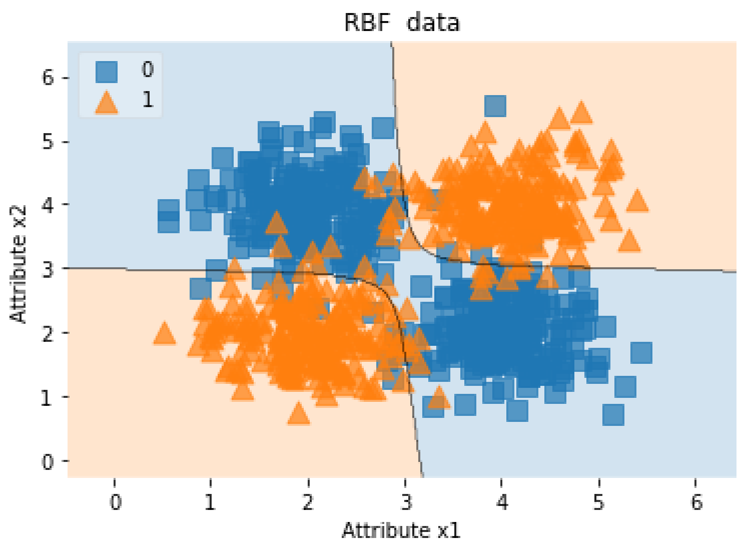
Answer: kernel gamma=1> kernel gamma=0.1> kernel gamma =0.01

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

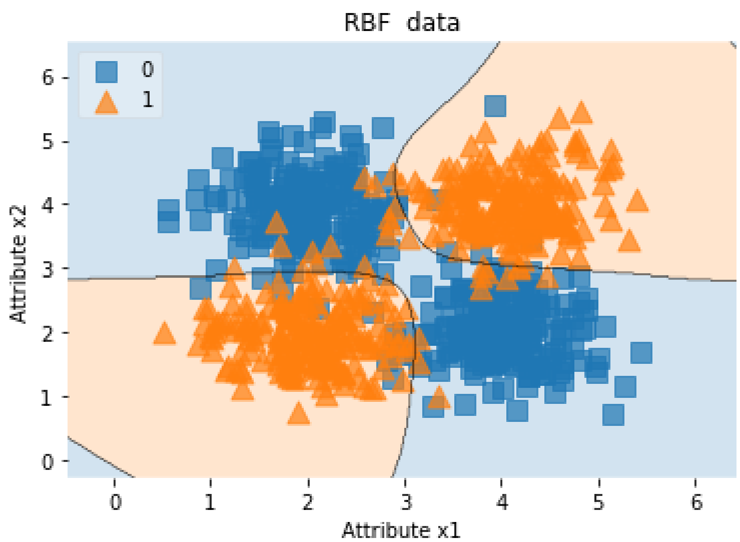
In [26]:
```
svm_rbf2_001.fit( Data2_X, Data2_Y)

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}

plot_decision_regions(X=Data2_X, y=Data2_Y, clf=svm_rbf2_001, 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('RBF  data')
plt.show()
```



In [28]:
```
svm_rbf2_01.fit( Data2_X, Data2_Y)

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}

plot_decision_regions(X=Data2_X, y=Data2_Y, clf=svm_rbf2_01, 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('RBF  data')
plt.show()
```



In [29]:
```
svm_rbf2_1.fit( Data2_X, Data2_Y)

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}

plot_decision_regions(X=Data2_X, y=Data2_Y, clf=svm_rbf2_1, 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('RBF  data')
plt.show()
```



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

Answer:

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:

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 [19]:
```
svm_rbf2_3 = SVC(C = 0.5, kernel='rbf', gamma=0.1)
svm_rbf_scores2_3 = cross_val_score(svm_rbf2_3, Data2_X, Data2_Y, cv=3, scoring='accuracy')
[svm_rbf_scores2_3, svm_rbf_scores2_3.mean()]
```

Out[19]:
```
[array([0.87313433, 0.93609023, 0.90225564]), 0.903826731006621]
```

In [20]:
```
svm_rbf2_4 = SVC(C = 0.5, kernel='rbf', gamma=0.1)
svm_rbf_scores2_4 = cross_val_score(svm_rbf2_4, Data2_X, Data2_Y, cv=4, scoring='accuracy')
[svm_rbf_scores2_4, svm_rbf_scores2_4.mean()]
```

Out[20]:
```
[array([0.91 , 0.92 , 0.895, 0.94 ]), 0.91625]
```

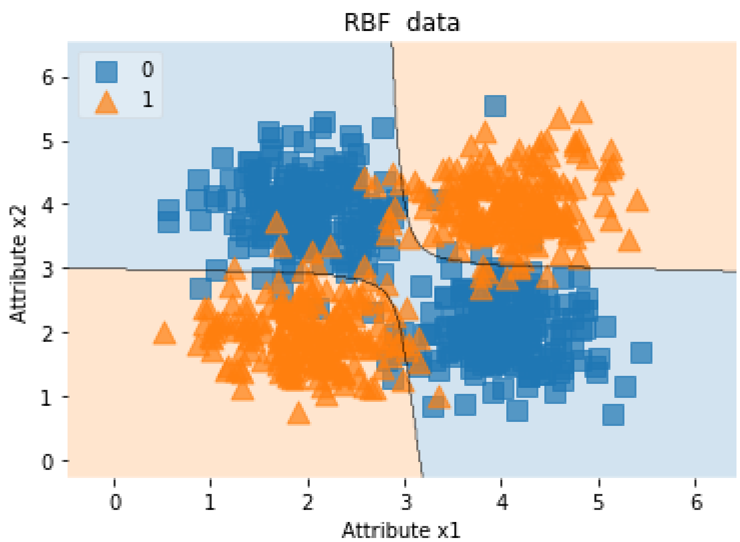
In [21]:
```
svm_rbf2_5 = SVC(C = 0.5, kernel='rbf', gamma=0.1)
svm_rbf_scores2_5 = cross_val_score(svm_rbf2_5, Data2_X, Data2_Y, cv=5, scoring='accuracy')
[svm_rbf_scores2_5, svm_rbf_scores2_5.mean()]
```

Out[21]:
```
[array([0.91875, 0.9    , 0.95   , 0.9125 , 0.95625]), 0.9275]
```

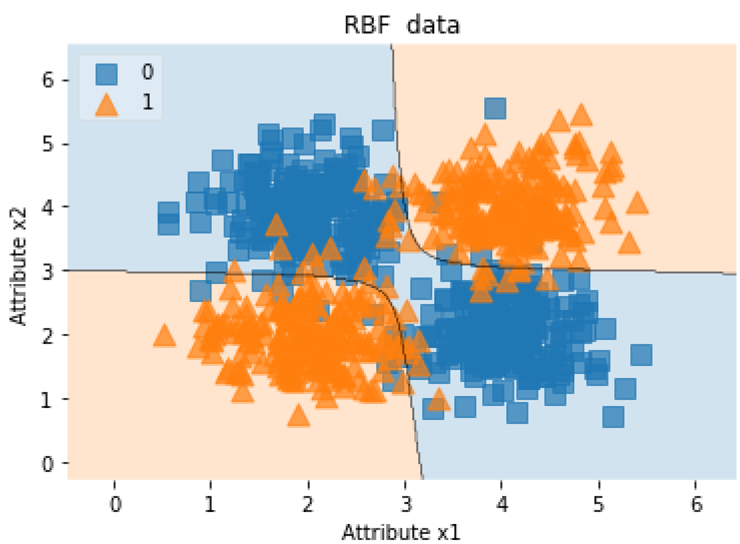
In [22]:
```
svm_rbf2_6 = SVC(C = 0.5, kernel='rbf', gamma=0.1)
svm_rbf_scores2_6 = cross_val_score(svm_rbf2_6, Data2_X, Data2_Y, cv=6, scoring='accuracy')
[svm_rbf_scores2_6, svm_rbf_scores2_6.mean()]
```

Out[22]:
```
[array([0.95522388, 0.89552239, 0.94776119, 0.91044776, 0.93939394,
        0.9469697 ]), 0.9325531433740389]
```

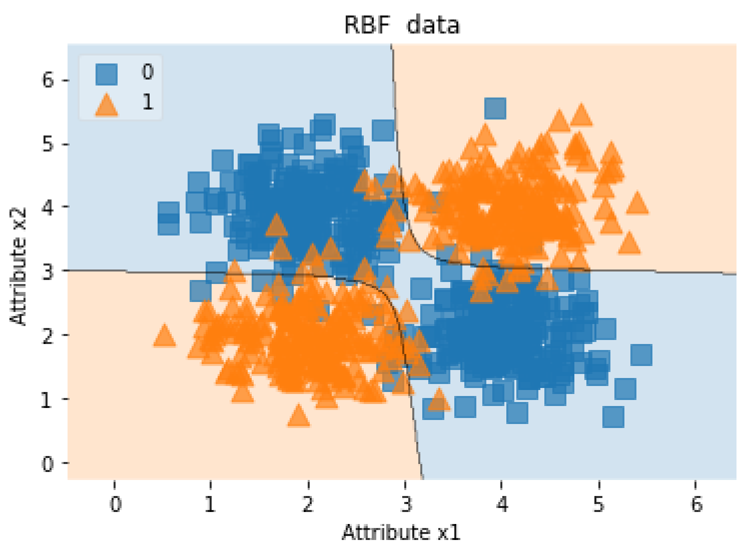
In [11]:
```
svm_rbf2_3.fit( Data2_X, Data2_Y)

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}

plot_decision_regions(X=Data2_X, y=Data2_Y, clf=svm_rbf2_3, 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('RBF  data')
plt.show()
```



In [12]:
```
svm_rbf2_4.fit( Data2_X, Data2_Y)

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}

plot_decision_regions(X=Data2_X, y=Data2_Y, clf=svm_rbf2_4, 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('RBF  data')
plt.show()
```



In [13]:
```
svm_rbf2_5.fit( Data2_X, Data2_Y)

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}

plot_decision_regions(X=Data2_X, y=Data2_Y, clf=svm_rbf2_5, 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('RBF  data')
plt.show()
```



Answer: The accuracy is increasing as the value of cv is increasing.

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 [23]:
```
dt3_4 = DecisionTreeClassifier(max_depth=4)
dt_scores3_4_accuracy = cross_val_score(dt3_4, Data3_X, Data3_Y, cv=30, scorin
g='accuracy')
dt_scores3_4_precision = cross_val_score(dt3_4, Data3_X, Data3_Y, cv=30, scori
ng='precision')
[dt_scores3_4_accuracy,dt_scores3_4_accuracy.mean(),dt_scores3_4_precision,dt_
scores3_4_precision.mean()]
```

Out[23]:
```
[array([0.92857143, 1.        , 1.        , 0.92857143, 1.        ,
        1.        , 1.        , 0.92857143, 1.        , 1.        ,
        0.92857143, 1.        , 1.        , 1.        , 1.        ,
        1.        , 1.        , 1.        , 0.85714286, 1.        ,
        0.91666667, 0.91666667, 1.        , 0.91666667, 0.91666667,
        1.        , 1.        , 1.        , 1.        , 0.91666667]),
 0.9718253968253968,
 array([1.        , 1.        , 1.        , 0.875     , 1.        ,
        1.        , 1.        , 0.875     , 1.        , 1.        ,
        1.        , 1.        , 1.        , 1.        , 1.        ,
        1.        , 1.        , 1.        , 0.77777778, 1.        ,
        0.85714286, 0.85714286, 1.        , 0.85714286, 0.85714286,
        1.        , 1.        , 1.        , 1.        , 0.85714286]),
 0.9604497354497356]
```

In [24]:
```
nb3 = GaussianNB()
nb3_accuracy = cross_val_score(nb3, Data3_X, Data3_Y, cv=30, scoring='accurac
y')
nb3_precision = cross_val_score(nb3, Data3_X, Data3_Y, cv=30, scoring='precisi
on')
[nb3_accuracy,nb3_accuracy.mean(),nb3_precision,nb3_precision.mean()]
```

Out[24]:
```
[array([1.        , 1.        , 0.92857143, 0.92857143, 1.        ,
        1.        , 1.        , 0.92857143, 1.        , 0.92857143,
        1.        , 1.        , 0.92857143, 1.        , 1.        ,
        1.        , 0.92857143, 1.        , 0.78571429, 1.        ,
        0.91666667, 0.83333333, 1.        , 0.91666667, 0.91666667,
        1.        , 1.        , 0.91666667, 1.        , 0.91666667]),
 0.9591269841269843,
 array([1.        , 1.        , 0.875     , 0.875     , 1.        ,
        1.        , 1.        , 0.875     , 1.        , 0.875     ,
        1.        , 1.        , 0.875     , 1.        , 1.        ,
        1.        , 0.875     , 1.        , 0.7       , 1.        ,
        0.85714286, 0.75      , 1.        , 0.85714286, 0.85714286,
        1.        , 1.        , 0.85714286, 1.        , 0.85714286]),
 0.9328571428571429]
```

In [25]:
```
knn3_1 = KNeighborsClassifier(n_neighbors=1)
knn3_1_accuracy = cross_val_score(knn3_1, Data3_X, Data3_Y, cv=30, scoring='ac
curacy')
knn3_1_precision = cross_val_score(knn3_1, Data3_X, Data3_Y, cv=30, scoring='p
recision')
[knn3_1_accuracy,knn3_1_accuracy.mean(),knn3_1_precision,knn3_1_precision.mean
()]
```

Out[25]:
```
[array([0.92857143, 1.        , 0.85714286, 0.92857143, 1.        ,
        1.        , 1.        , 0.92857143, 0.92857143, 1.        ,
        1.        , 1.        , 1.        , 0.92857143, 1.        ,
        1.        , 1.        , 1.        , 0.64285714, 1.        ,
        0.91666667, 0.75      , 1.        , 0.83333333, 0.91666667,
        0.91666667, 1.        , 0.91666667, 1.        , 0.83333333]),
 0.9408730158730161,
 array([0.875     , 1.        , 0.85714286, 0.875     , 1.        ,
        1.        , 1.        , 0.875     , 1.        , 1.        ,
        1.        , 1.        , 1.        , 0.875     , 1.        ,
        1.        , 1.        , 1.        , 0.6       , 1.        ,
        0.85714286, 0.71428571, 1.        , 0.83333333, 0.85714286,
        1.        , 1.        , 0.85714286, 1.        , 0.83333333]),
 0.9303174603174603]
```

In [26]:
```
svm_linear3 = SVC(C=0.5, kernel='linear')
svm_linear3_accuracy = cross_val_score(svm_linear3, Data3_X, Data3_Y, cv=30, s
coring='accuracy')
svm_linear3_precision = cross_val_score(dt3_4, Data3_X, Data3_Y, cv=30, scorin
g='precision')
[svm_linear3_accuracy,svm_linear3_accuracy.mean(),svm_linear3_precision,svm_li
near3_precision.mean()]
```

Out[26]:
```
[array([0.57142857, 0.71428571, 0.5       , 0.64285714, 0.64285714,
        0.64285714, 0.71428571, 0.5       , 0.71428571, 0.5       ,
        0.64285714, 0.64285714, 0.57142857, 0.71428571, 0.78571429,
        0.64285714, 0.64285714, 0.78571429, 0.5       , 0.71428571,
        0.58333333, 0.66666667, 0.75      , 0.66666667, 0.66666667,
        0.66666667, 0.58333333, 0.58333333, 0.66666667, 0.66666667]),
 0.6428571428571429,
 array([1.        , 1.        , 1.        , 0.875     , 1.        ,
        1.        , 1.        , 0.875     , 1.        , 1.        ,
        1.        , 1.        , 1.        , 1.        , 1.        ,
        1.        , 1.        , 1.        , 0.77777778, 1.        ,
        0.85714286, 0.85714286, 1.        , 0.85714286, 0.85714286,
        1.        , 1.        , 1.        , 1.        , 0.85714286]),
 0.9604497354497356]
```

In [47]:
```
svm_poly3_3 = SVC(C=0.5, kernel='poly',degree=3, gamma = 'auto')
svm_poly_scores3_3 = cross_val_score(svm_poly3_3, Data3_X, Data3_Y, cv=30, sco
ring='accuracy')
svm_poly3_1_precision = cross_val_score(svm_poly3_1, Data3_X, Data3_Y, cv=30,
scoring='precision')
[svm_poly3_1_accuracy,svm_poly3_1_accuracy.mean(),svm_poly3_1_precision,svm_po
ly3_1_precision.mean()]
```

Out[47]:
```
[array([0.92857143, 1.        , 0.78571429, 0.85714286, 0.71428571,
        0.85714286, 0.78571429, 0.78571429, 0.85714286, 0.85714286,
        0.85714286, 0.85714286, 0.78571429, 0.85714286, 0.85714286,
        0.92857143, 0.71428571, 0.92857143, 0.78571429, 1.        ,
        0.83333333, 0.83333333, 0.91666667, 0.83333333, 0.83333333,
        1.        , 0.91666667, 0.83333333, 0.75      , 0.91666667]),
 0.8555555555555555,
 array([0.875     , 1.        , 0.75      , 0.77777778, 0.63636364,
        0.77777778, 0.7       , 0.7       , 0.77777778, 0.85714286,
        0.77777778, 0.77777778, 0.7       , 0.77777778, 0.77777778,
        0.875     , 0.66666667, 0.875     , 0.7       , 1.        ,
        0.75      , 0.75      , 0.85714286, 0.75      , 0.75      ,
        1.        , 0.85714286, 0.83333333, 0.66666667, 0.85714286]),
 0.7950348725348725]
```

In [31]:
```
svm_rbf3_001 = SVC(C = 0.5, kernel='rbf', gamma=0.01)
dt_scores3_4_accuracy = cross_val_score(dt3_4, Data3_X, Data3_Y, cv=30, scorin
g='accuracy')
dt_scores3_4_precision = cross_val_score(dt3_4, Data3_X, Data3_Y, cv=30, scori
ng='precision')
[dt_scores3_4_accuracy,dt_scores3_4_accuracy.mean(),dt_scores3_4_precision,dt_
scores3_4_precision.mean()]
```

Out[31]:
```
[array([0.92857143, 1.        , 1.        , 0.92857143, 1.        ,
        1.        , 1.        , 0.92857143, 1.        , 1.        ,
        0.92857143, 1.        , 1.        , 1.        , 1.        ,
        1.        , 1.        , 1.        , 0.85714286, 1.        ,
        0.91666667, 0.91666667, 1.        , 0.91666667, 0.91666667,
        1.        , 1.        , 1.        , 1.        , 0.91666667]),
 0.9718253968253968,
 array([1.        , 1.        , 1.        , 0.875     , 1.        ,
        1.        , 1.        , 0.875     , 1.        , 1.        ,
        1.        , 1.        , 1.        , 1.        , 1.        ,
        1.        , 1.        , 1.        , 0.77777778, 1.        ,
        0.85714286, 0.85714286, 1.        , 0.85714286, 0.85714286,
        1.        , 1.        , 1.        , 1.        , 0.85714286]),
 0.9604497354497356]
```

Answer:

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 [7]:
```
svm_linear3 = SVC(C=0.5, kernel='linear')
n_est_list = [1,2,3,4,5,10,20]
for n_est in n_est_list:
    bagging = BaggingClassifier(base_estimator=svm_linear3, n_estimators=n_est
)
    scores = cross_val_score(bagging, Data3_X, Data3_Y, cv=5, scoring='accurac
y')
    print("Bagging Accuracy: %.2f standard deviation : +/- %.2f #estimators: %
d" % (scores.mean(), scores.std(), n_est))
```

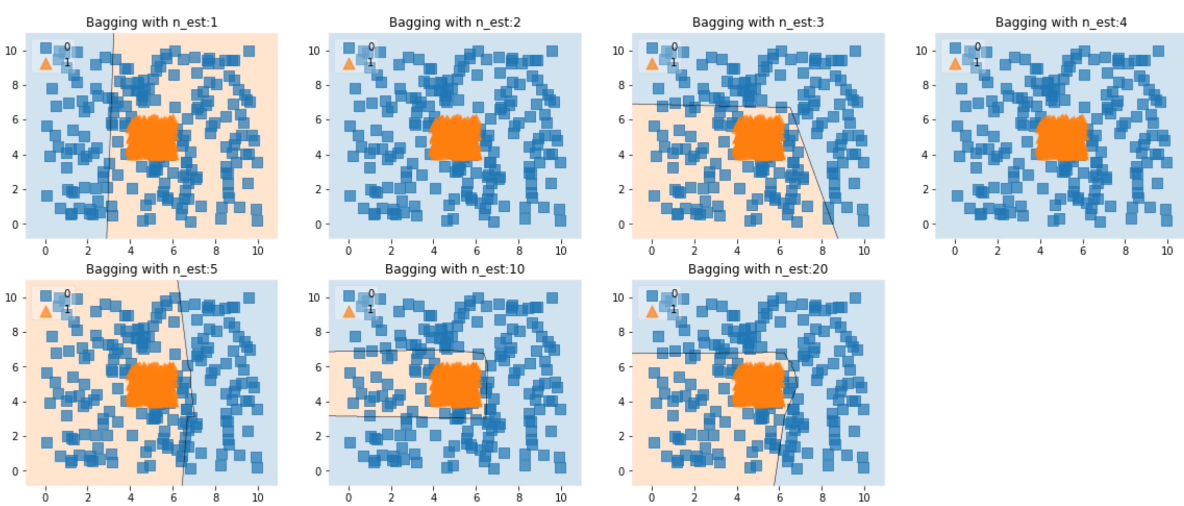
```
Bagging Accuracy: 0.57 standard deviation : +/- 0.06 #estimators: 1
Bagging Accuracy: 0.61 standard deviation : +/- 0.10 #estimators: 2
Bagging Accuracy: 0.57 standard deviation : +/- 0.07 #estimators: 3
Bagging Accuracy: 0.57 standard deviation : +/- 0.09 #estimators: 4
Bagging Accuracy: 0.60 standard deviation : +/- 0.06 #estimators: 5
Bagging Accuracy: 0.68 standard deviation : +/- 0.06 #estimators: 10
Bagging Accuracy: 0.68 standard deviation : +/- 0.09 #estimators: 20
```

Answer:

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

In [14]:
```
fig = plt.figure(figsize=(20, 8))
count = 0;


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;
    bagging = BaggingClassifier(base_estimator=svm_linear3, n_estimators=n_est
)
    bagging.fit(Data3_X, Data3_Y)
    ax = plt.subplot(2,4,count)
    fig = 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.title('Bagging with n_est:'+str(n_est))

plt.show()
```

```
/users/PES0801/shravreddy/.local/lib/python3.6/site-packages/mlxtend/plottin
g/decision_regions.py:247: UserWarning: No contour levels were found within t
he data range.
  antialiased=True)
/users/PES0801/shravreddy/.local/lib/python3.6/site-packages/mlxtend/plottin
g/decision_regions.py:247: UserWarning: No contour levels were found within t
he data range.
  antialiased=True)
```



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:

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.

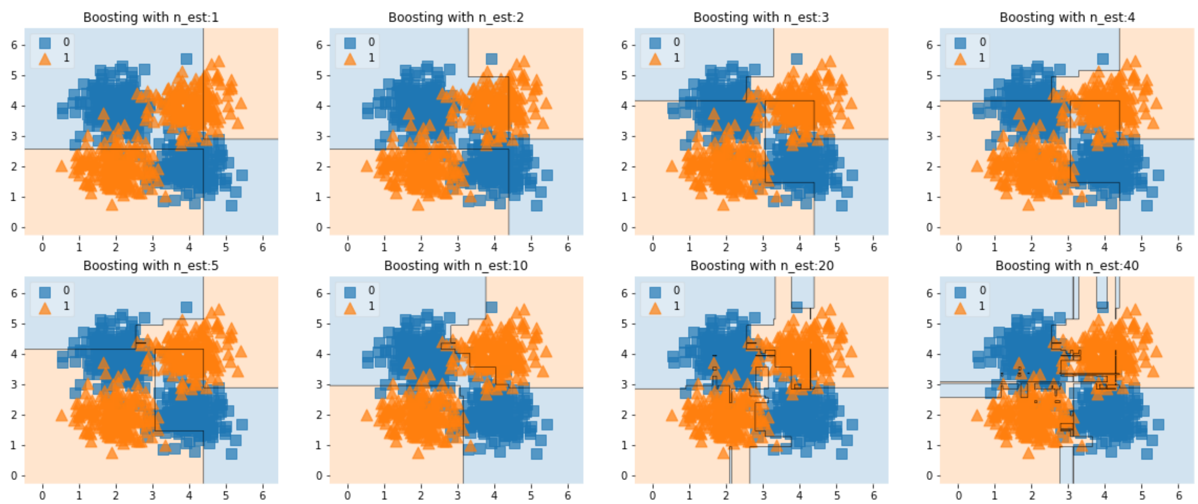
In [15]:
```
dt = DecisionTreeClassifier(max_depth=2)
n_est_list = [1,2,3,4,5,10,20,40]
for n_est in n_est_list:
    # create an instance of a boosting classifier with 'n_est' estimators
    boosting = AdaBoostClassifier(base_estimator=dt, n_estimators=n_est)
    # compute cross-validation accuracy for each bagging classifier
    scores = cross_val_score(boosting, Data2_X, Data2_Y, cv=10, scoring='accur
acy')
    print("Boosting Accuracy: %.2f standsard deviation +/- %.2f #estimators: %
d" % (scores.mean(), scores.std(), n_est))
```

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

Answer:

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

In [16]:
```
fig = plt.figure(figsize=(20, 8))
count = 0;
for n_est in n_est_list:
    count = count + 1;
    boosting = AdaBoostClassifier(base_estimator=dt, n_estimators=n_est)
    boosting.fit(Data2_X, Data2_Y)
    ax = plt.subplot(2,4,count)
    fig = 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.title('Boosting with n_est:'+str(n_est))

plt.show()
```



Answer:

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 [7]:
```
from sklearn import datasets
cancer = datasets.load_breast_cancer()
```

The features are:

In [8]:
```
cancer.feature_names
```

Out[8]:
```
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 [9]:
```
cancer.target_names
```

Out[9]:
```
array(['malignant', 'benign'], dtype='<U9')
```

Create dataset for classification

In [10]:
```
X = cancer.data
Y = cancer.target
```

Number of samples are:

In [11]:
```
X.shape
```

Out[11]:
```
(569, 30)
```

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 [16]:
```
svm_linear_0001 = SVC(C=0.001, kernel='linear')
svm_linear_scores_0001 = cross_val_score(svm_linear_0001, X, Y, cv=10, scoring
='accuracy')
[svm_linear_scores_0001, svm_linear_scores_0001.mean()]
```

Out[16]:
```
[array([0.9137931 , 0.9137931 , 0.94736842, 0.94736842, 1.        ,
        0.98245614, 0.92982456, 0.89285714, 0.92857143, 0.94642857]),
 0.9402460893613342]
```

In [17]:
```
svm_linear_001 = SVC(C=0.01, kernel='linear')
svm_linear_scores_001 = cross_val_score(svm_linear_001, X, Y, cv=10, scoring=
'accuracy')
[svm_linear_scores_001, svm_linear_scores_001.mean()]
```

Out[17]:
```
[array([0.96551724, 0.9137931 , 0.94736842, 0.94736842, 1.        ,
        0.96491228, 0.92982456, 0.89285714, 0.96428571, 0.94642857]),
 0.947235545760954]
```

In [18]:
```
svm_linear_01 = SVC(C=0.1, kernel='linear')
svm_linear_scores_01 = cross_val_score(svm_linear_01, X, Y, cv=10, scoring='ac
curacy')
[svm_linear_scores_01, svm_linear_scores_01.mean()]
```

Out[18]:
```
[array([0.96551724, 0.93103448, 0.92982456, 0.94736842, 0.98245614,
        0.96491228, 0.92982456, 0.91071429, 0.96428571, 0.94642857]),
 0.9472366260478783]
```

In [19]:
```
svm_linear_1 = SVC(C=1, kernel='linear')
svm_linear_scores_1 = cross_val_score(svm_linear_1, X, Y, cv=10, scoring='accu
racy')
[svm_linear_scores_1, svm_linear_scores_1.mean()]
```

Out[19]:
```
[array([0.98275862, 0.93103448, 0.92982456, 0.94736842, 0.96491228,
        0.98245614, 0.92982456, 0.94642857, 0.96428571, 0.96428571]),
 0.9543179068360554]
```

In [22]:
```
#Polynomial varying C and degree
svm_poly1_0001 = SVC(C=0.001, kernel='poly',degree=1, gamma = 'auto')
svm_poly_scores1_0001 = cross_val_score(svm_poly1_0001, X, Y, cv=10, scoring=
'accuracy')
[svm_poly_scores1_0001, svm_poly_scores1_0001.mean()]
```

Out[22]:
```
[array([0.9137931 , 0.89655172, 0.9122807 , 0.94736842, 0.94736842,
        0.89473684, 0.98245614, 0.92857143, 0.89285714, 0.96428571]),
 0.9280269639616281]
```

In [23]:
```
svm_poly1_001 = SVC(C=0.01, kernel='poly',degree=1, gamma = 'auto')
svm_poly_scores1_001 = cross_val_score(svm_poly1_001, X, Y, cv=10, scoring='ac
curacy')
[svm_poly_scores1_001, svm_poly_scores1_001.mean()]
```

Out[23]:
```
[array([0.93103448, 0.9137931 , 0.94736842, 0.92982456, 0.98245614,
        0.92982456, 0.96491228, 0.91071429, 0.92857143, 0.96428571]),
 0.9402784979690605]
```

In [24]:
```
svm_poly1_01 = SVC(C=0.1, kernel='poly',degree=1, gamma = 'auto')
svm_poly_scores1_01 = cross_val_score(svm_poly1_01, X, Y, cv=10, scoring='accu
racy')
[svm_poly_scores1_01, svm_poly_scores1_01.mean()]
```

Out[24]:
```
[array([0.96551724, 0.9137931 , 0.94736842, 0.94736842, 1.        ,
        0.94736842, 0.92982456, 0.91071429, 0.96428571, 0.94642857]),
 0.947266874081756]
```

In [25]:
```
svm_poly1_1 = SVC(C=1, kernel='poly',degree=1, gamma = 'auto')
svm_poly_scores1_1 = cross_val_score(svm_poly1_1, X, Y, cv=10, scoring='accura
cy')
[svm_poly_scores1_1, svm_poly_scores1_1.mean()]
```

Out[25]:
```
[array([0.96551724, 0.9137931 , 0.94736842, 0.94736842, 0.98245614,
        0.96491228, 0.92982456, 0.91071429, 0.96428571, 0.94642857]),
 0.947266874081756]
```

In [26]:
```
svm_poly2_0001 = SVC(C=0.001, kernel='poly',degree=2, gamma = 'auto')
svm_poly_scores2_0001 = cross_val_score(svm_poly2_0001, X, Y, cv=10, scoring=
'accuracy')
[svm_poly_scores2_0001, svm_poly_scores2_0001.mean()]
```

Out[26]:
```
[array([0.98275862, 0.9137931 , 0.9122807 , 0.92982456, 0.96491228,
        0.98245614, 0.92982456, 0.94642857, 0.96428571, 0.96428571]),
 0.9490849969751964]
```

In [27]:
```
svm_poly2_001 = SVC(C=0.01, kernel='poly',degree=2, gamma = 'auto')
svm_poly_scores2_001 = cross_val_score(svm_poly2_001, X, Y, cv=10, scoring='ac
curacy')
[svm_poly_scores2_001, svm_poly_scores2_001.mean()]
```

Out[27]:
```
[array([0.98275862, 0.9137931 , 0.9122807 , 0.96491228, 0.96491228,
        0.98245614, 0.94736842, 0.94642857, 1.        , 0.96428571]),
 0.957919583441362]
```

In [28]:
```
svm_poly2_01 = SVC(C=0.1, kernel='poly',degree=2, gamma = 'auto')
svm_poly_scores2_01 = cross_val_score(svm_poly2_01, X, Y, cv=10, scoring='accu
racy')
[svm_poly_scores2_01, svm_poly_scores2_01.mean()]
```

Out[28]:
```
[array([0.98275862, 0.9137931 , 0.9122807 , 0.96491228, 0.96491228,
        0.98245614, 0.94736842, 0.96428571, 0.98214286, 0.96428571]),
 0.957919583441362]
```

In [29]:
```
svm_poly2_1 = SVC(C=1, kernel='poly',degree=2, gamma = 'auto')
svm_poly_scores2_1 = cross_val_score(svm_poly2_1, X, Y, cv=10, scoring='accura
cy')
[svm_poly_scores2_1, svm_poly_scores2_1.mean()]
```

Out[29]:
```
[array([0.98275862, 0.9137931 , 0.92982456, 0.96491228, 0.96491228,
        0.94736842, 0.94736842, 0.96428571, 1.        , 0.96428571]),
 0.9579509117621638]
```

In [ ]:
```
#RBF varying C and gamma
svm_rbf2_0001 = SVC(C = 1, kernel='rbf', gamma=0.001)
svm_rbf_scores2_0001 = cross_val_score(svm_rbf2_0001, X, Y, cv=10, scoring='ac
curacy')
[svm_rbf_scores2_0001, svm_rbf_scores2_0001.mean()]
```

In [ ]:
```
svm_rbf2_001 = SVC(C = 1, kernel='rbf', gamma=0.01)
svm_rbf_scores2_001 = cross_val_score(svm_rbf2_001, X, Y, cv=10, scoring='accuracy')
[svm_rbf_scores2_001, svm_rbf_scores2_001.mean()]
```

In [ ]:
```
svm_rbf2_01 = SVC(C = 1, kernel='rbf', gamma=0.1)
svm_rbf_scores2_01 = cross_val_score(svm_rbf2_01, X, Y, cv=10, scoring='accuracy')
[svm_rbf_scores2_01, svm_rbf_scores2_01.mean()]
```

In [ ]:
```
svm_rbf2_1 = SVC(C = 1, kernel='rbf', gamma=0.001)
svm_rbf_scores2_1 = cross_val_score(svm_rbf2_1, X, Y, cv=10, scoring='accuracy')
[svm_rbf_scores2_1, svm_rbf_scores2_1.mean()]
```

Answer:

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

Answer:

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: