In [ ]:
```python
#preprocess the data
#select the necessary libraries
import numpy as np
import matplotlib.pyplot as plt
from matplotlib import style
style.use("ggplot")
from sklearn.cluster import KMeans
import pandas as pd
from sklearn import preprocessing
from sklearn.preprocessing import Normalizer
from sklearn.decomposition import PCA
from sklearn.metrics import accuracy_score
```

In [2]:
```python
#select the data source
#the target data
path='D:/coding/divorce.csv'
Data=pd.read_csv(path)
print("Selected Data Source:\n",Data)
```

```
Selected Data Source:
      Atr1  Atr2  Atr3  Atr4  Atr5  Atr6  Atr7  Atr8  Atr9  Atr10  ...  Atr46  \
0       2     2     4     1     0     0     0     0     0      0  ...      2
1       4     4     4     4     4     0     0     4     4      4  ...      2
2       2     2     2     2     1     3     2     1     1      2  ...      3
3       3     2     3     2     3     3     3     3     3      3  ...      2
4       2     2     1     1     1     1     0     0     0      0  ...      2
..    ...   ...   ...   ...   ...   ...   ...   ...   ...    ...  ...    ...
165     0     0     0     0     0     0     0     0     0      0  ...      1
166     0     0     0     0     0     0     0     0     0      0  ...      4
167     1     1     0     0     0     0     0     0     0      1  ...      3
168     0     0     0     0     0     0     0     0     0      0  ...      3
169     0     0     0     0     0     0     0     1     0      0  ...      3

     Atr47  Atr48  Atr49  Atr50  Atr51  Atr52  Atr53  Atr54  Class
0        1      3      3      3      2      3      2      1      1
1        2      3      4      4      4      4      2      2      1
2        2      3      1      1      1      2      2      2      1
3        2      3      3      3      3      2      2      2      1
4        1      2      3      2      2      2      1      0      1
..     ...    ...    ...    ...    ...    ...    ...    ...    ...
165      0      4      1      1      4      2      2      2      0
166      1      2      2      2      2      3      2      2      0
167      0      2      0      1      1      3      0      0      0
168      3      2      2      3      2      4      3      1      0
169      4      4      0      1      3      3      3      1      0

[170 rows x 55 columns]
```

In [3]:
```python
#Transform the target to numeric array
Encoder=preprocessing.LabelEncoder()
Encoded_Data=Data.apply(preprocessing.LabelEncoder().fit_transform)
print("Transformed Data:\n",Encoded_Data)
Numeric_Array=Encoded_Data.values
print("Numeric Array\n", Numeric_Array)
```

```
Transformed Data:
      Atr1  Atr2  Atr3  Atr4  Atr5  Atr6  Atr7  Atr8  Atr9  Atr10  ...  Atr46  \
0       2     2     4     1     0     0     0     0     0      0  ...      2
1       4     4     4     4     4     0     0     4     4      4  ...      2
2       2     2     2     2     1     3     2     1     1      2  ...      3
3       3     2     3     2     3     3     3     3     3      3  ...      2
4       2     2     1     1     1     1     0     0     0      0  ...      2
..    ...   ...   ...   ...   ...   ...   ...   ...   ...    ...  ...    ...
165     0     0     0     0     0     0     0     0     0      0  ...      1
166     0     0     0     0     0     0     0     0     0      0  ...      4
167     1     1     0     0     0     0     0     0     0      1  ...      3
168     0     0     0     0     0     0     0     0     0      0  ...      3
169     0     0     0     0     0     0     0     1     0      0  ...      3

     Atr47  Atr48  Atr49  Atr50  Atr51  Atr52  Atr53  Atr54  Class
0        1      3      3      3      2      3      2      1      1
1        2      3      4      4      4      4      2      2      1
2        2      3      1      1      1      2      2      2      1
3        2      3      3      3      3      2      2      2      1
4        1      2      3      2      2      2      1      0      1
..     ...    ...    ...    ...    ...    ...    ...    ...    ...
165      0      4      1      1      4      2      2      2      0
166      1      2      2      2      2      3      2      2      0
167      0      2      0      1      1      3      0      0      0
168      3      2      2      3      2      4      3      1      0
169      4      4      0      1      3      3      3      1      0

[170 rows x 55 columns]
Numeric Array
 [[2 2 4 ... 2 1 1]
 [4 4 4 ... 2 2 1]
 [2 2 2 ... 2 2 1]
 ...
 [1 1 0 ... 0 0 0]
 [0 0 0 ... 3 1 0]
 [0 0 0 ... 3 1 0]]
```

In [4]:
```python
#select test sample
split=10
Training_Sample=Numeric_Array[:split,:]
print(("Training Sample:\n", Training_Sample))
```

```
('Training Sample:\n', array([[2, 2, 4, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 1, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 1, 1, 2, 1, 2, 0, 1, 2, 1, 3, 3, 2, 1, 1, 2,
        3, 2, 1, 3, 3, 3, 2, 3, 2, 1, 1],
       [4, 4, 4, 4, 4, 0, 0, 4, 4, 4, 4, 3, 4, 0, 4, 4, 4, 4, 3, 2, 1, 1,
        0, 2, 2, 1, 2, 0, 1, 1, 0, 4, 2, 3, 0, 2, 3, 4, 2, 4, 2, 2, 3, 4,
        2, 2, 2, 3, 4, 4, 4, 4, 2, 2, 1],
       [2, 2, 2, 2, 1, 3, 2, 1, 1, 2, 3, 4, 2, 3, 3, 3, 3, 3, 3, 2, 1, 0,
        1, 2, 2, 2, 2, 2, 3, 2, 3, 3, 1, 1, 1, 1, 2, 1, 3, 3, 3, 3, 2, 3,
        2, 3, 2, 3, 1, 1, 1, 2, 2, 2, 1],
       [3, 2, 3, 2, 3, 3, 3, 3, 3, 3, 4, 3, 3, 4, 3, 3, 3, 3, 3, 4, 1, 1,
        1, 1, 2, 1, 1, 1, 1, 3, 2, 3, 2, 2, 1, 1, 3, 3, 4, 4, 2, 2, 3, 2,
        3, 2, 2, 3, 3, 3, 3, 2, 2, 2, 1],
       [2, 2, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 1, 2, 1, 1, 0,
        0, 0, 0, 2, 1, 2, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 2, 1, 0, 2, 3, 0,
        2, 2, 1, 2, 3, 2, 2, 2, 1, 0, 1],
       [0, 0, 1, 0, 0, 2, 0, 0, 0, 1, 0, 2, 1, 0, 2, 0, 2, 1, 0, 1, 0, 0,
        0, 0, 2, 2, 0, 0, 0, 0, 4, 1, 1, 1, 1, 1, 1, 2, 0, 2, 2, 1, 2, 3,
        0, 2, 2, 1, 2, 1, 1, 1, 2, 0, 1],
       [3, 3, 3, 2, 1, 3, 4, 3, 2, 2, 2, 2, 2, 3, 2, 3, 3, 3, 3, 2, 3, 3,
        3, 3, 2, 3, 3, 2, 2, 2, 1, 2, 2, 1, 1, 2, 3, 2, 2, 3, 3, 3, 3, 4,
        3, 3, 2, 3, 2, 3, 3, 2, 2, 2, 1],
       [2, 1, 2, 2, 2, 1, 0, 3, 3, 2, 4, 3, 2, 3, 4, 3, 2, 3, 2, 1, 2, 1,
        1, 2, 3, 3, 2, 2, 2, 3, 1, 1, 0, 2, 2, 1, 4, 4, 4, 4, 4, 4, 3, 2,
        0, 0, 1, 2, 2, 2, 1, 1, 1, 0, 1],
       [2, 2, 1, 0, 0, 4, 1, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 2, 2,
        2, 3, 2, 3, 2, 3, 2, 3, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2,
        2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1],
       [1, 1, 1, 1, 1, 2, 0, 2, 2, 2, 3, 0, 0, 2, 1, 0, 1, 2, 1, 0, 0, 0,
        0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 2, 2, 1, 2, 3, 2,
        2, 2, 0, 2, 2, 2, 2, 4, 3, 3, 1]], dtype=int64))
```

In [5]:
```python
#select test sample
Test_Sample=Numeric_Array[split:,:]
print("Test Sample:\n", Test_Sample)
```

```
Test Sample:
 [[4 4 4 ... 4 4 1]
 [4 4 4 ... 4 4 1]
 [3 4 3 ... 4 4 1]
 ...
 [1 1 0 ... 0 0 0]
 [0 0 0 ... 3 1 0]
 [0 0 0 ... 3 1 0]]
```

In [6]:
```python
#select input attributes from training sample
XTrain_Sample=Training_Sample[:,:-1]
print("Input(Indipendent) Attributes of training sample\n", XTrain_Sample)
```

```
Input(Indipendent) Attributes of training sample
 [[2 2 4 1 0 0 0 0 0 0 1 0 1 1 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 1 1 2 1 2 0 1
  2 1 3 3 2 1 1 2 3 2 1 3 3 3 2 3 2 1]
 [4 4 4 4 4 0 0 4 4 4 4 3 4 0 4 4 4 4 3 2 1 1 0 2 2 1 2 0 1 1 0 4 2 3 0 2
  3 4 2 4 2 2 3 4 2 2 2 3 4 4 4 4 2 2]
 [2 2 2 2 1 3 2 1 1 2 3 4 2 3 3 3 3 3 3 2 1 0 1 2 2 2 2 2 3 2 3 3 1 1 1 1
  2 1 3 3 3 3 2 3 2 3 2 3 1 1 1 2 2 2]
 [3 2 3 2 3 3 3 3 3 3 4 3 3 4 3 3 3 3 3 4 1 1 1 1 2 1 1 1 1 3 2 3 2 2 1 1
  3 3 4 4 2 2 3 2 3 2 2 3 3 3 3 2 2 2]
 [2 2 1 1 1 1 0 0 0 0 0 1 0 1 1 1 1 1 2 1 1 0 0 0 0 2 1 2 1 1 1 1 1 1 0 0
  0 0 2 1 0 2 3 0 2 2 1 2 3 2 2 2 1 0]
 [0 0 1 0 0 2 0 0 0 1 0 2 1 0 2 0 2 1 0 1 0 0 0 0 2 2 0 0 0 0 4 1 1 1 1 1
  1 2 0 2 2 1 2 3 0 2 2 1 2 1 1 1 2 0]
 [3 3 3 2 1 3 4 3 2 2 2 2 2 3 2 3 3 3 3 2 3 3 3 3 2 3 3 2 2 2 1 2 2 1 1 2
  3 2 2 3 3 3 3 4 3 3 2 3 2 3 3 2 2 2]
 [2 1 2 2 2 1 0 3 3 2 4 3 2 3 4 3 2 3 2 1 2 1 1 2 3 3 2 2 2 3 1 1 0 2 2 1
  4 4 4 4 4 4 3 2 0 0 1 2 2 2 1 1 1 0]
 [2 2 1 0 0 4 1 3 3 3 3 3 3 3 3 3 3 3 3 3 2 2 2 3 2 3 2 3 2 3 1 1 1 1 1 1
  1 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1]
 [1 1 1 1 1 2 0 2 2 2 3 0 0 2 1 0 1 2 1 0 0 0 0 1 1 1 1 1 1 1 1 1 0 1 0 0
  1 1 2 2 1 2 3 2 2 2 0 2 2 2 2 4 3 3]]
```

In [7]:
```python
#select output attribute from training sample
YTrain_Sample=Training_Sample[:,-1]
print("Ouput(Dependent) Attributes of training sample\n", YTrain_Sample)
```

```
Ouput(Dependent) Attributes of training sample
 [1 1 1 1 1 1 1 1 1 1]
```

In [8]:
```python
#select input attributes from test sample
XTest_Sample=Test_Sample[:,:-1]
print("Indipendent Attributes of test sample\n\n", XTest_Sample)
```

```
Indipendent Attributes of test sample

 [[4 4 4 ... 4 4 4]
 [4 4 4 ... 4 4 4]
 [3 4 3 ... 4 4 4]
 ...
 [1 1 0 ... 3 0 0]
 [0 0 0 ... 4 3 1]
 [0 0 0 ... 3 3 1]]
```

In [9]:
```python
#select output attributes from test sample
Actual_YTest_Sample=Test_Sample[:,-1]
print("Dependent Attributes of test sample\n\n", Actual_YTest_Sample)
```

```
Dependent Attributes of test sample

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

In [10]:
```python
#Compress input attributes data training sample into two attributes
# Defining PCA object for compressing the attributes
pca = PCA(n_components =2)
#use PCA objects to compress the input attributes of the Training Sample
Compressed_XTrain_Sample = pca.fit_transform(XTrain_Sample)
print("Compressed input atributes of Training Sample:\n",Compressed_XTrain_Sample)
```

```
Compressed input atributes of Training Sample:
 [[ 6.36181353  4.35778488]
 [-6.9591194   6.97974282]
 [-1.86934224 -2.66948614]
 [-5.13634632  1.04824906]
 [ 6.75631959  0.08437835]
 [ 7.1378128  -1.64959095]
 [-4.48735944 -1.40709007]
 [-3.66241026 -2.41900773]
 [-2.47626135 -5.28877   ]
 [ 4.33489309  0.9637898 ]]
```

In [11]:
```python
Compressed_XTest_Sample = pca.fit_transform(XTest_Sample)
print("Compressed input atributes of Training Sample:\n",Compressed_XTest_Sample)
```

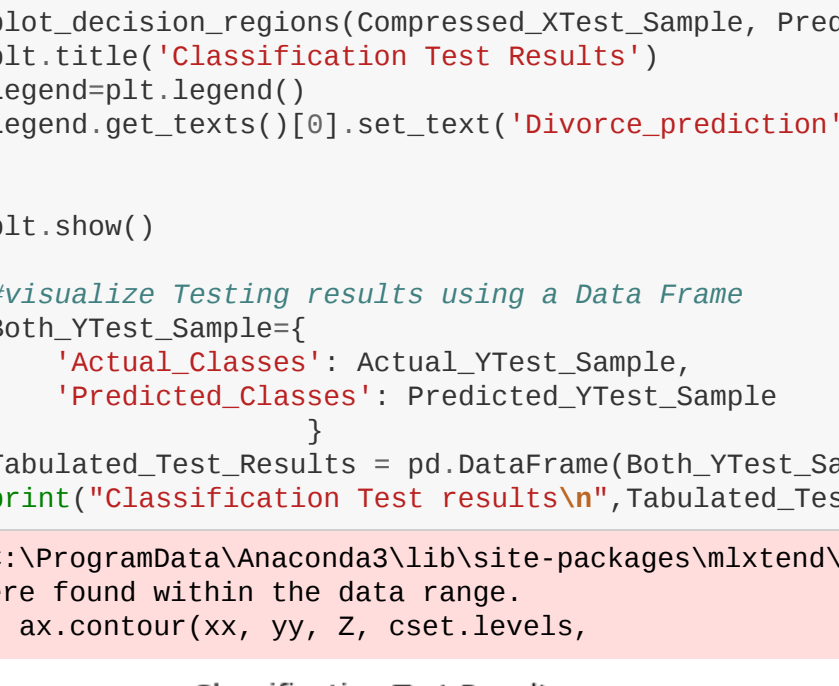
```
Compressed input atributes of Training Sample:
 [[ 15.01521765   0.2461208 ]
 [ 15.01521765   0.2461208 ]
 [ 13.68871096   0.40720336]
 [ 13.68871096   0.40720336]
 [ 13.54125871   0.55312961]
 [ 13.68998274  -0.62677004]
 [ 13.68998274  -0.62677004]
 [ 13.57965234   1.23662623]
 [ 13.14518801  -0.26826504]
 [ 13.57965234   1.23662623]
 [ 13.12497241  -1.06981971]
 [ 13.12497241  -1.06981971]
 [ 12.8316997    2.11120323]
 [ 12.47182686  -0.5885011 ]
 [ 12.55996208  -1.51286938]
 [ 12.47182686  -0.5885011 ]
 [ 12.39850714  -0.42766145]
 [ 12.37701977  -0.19524271]
 [ 12.38496343  -0.69810622]
 [ 12.47309865  -1.6224745 ]
 [ 12.38496343  -0.69810622]
 [ 12.36276825   0.24092177]
 [ 12.34128087   0.4733405 ]
 [ 12.26866896  -0.0724291 ]
 [ 12.27463303   1.16529005]
 [ 11.8199531   -1.14115589]
 [ 11.87103832  -0.35105671]
 [ 11.64063063   0.40315851]
 [ 11.64063063   0.40315851]
 [ 11.68940711  -0.07931429]
 [ 11.16013567  -1.19094717]
 [ 11.25621455  -2.61817896]
 [ 11.27682954   1.43217611]
 [ 11.16807933  -1.69381068]
 [ 11.25621455  -2.61817896]
 [ 11.14659196  -1.46139194]
 [ 11.16807933  -1.69381068]
 [ 11.14659196  -1.46139194]
 [ 11.25621455  -2.61817896]
 [ 11.16807933  -1.69381068]
 [ 11.25621455  -2.61817896]
 [ 11.07327224  -1.30055229]
 [ 11.01604597  -0.39955035]
 [ 10.71181921   0.98912643]
 [ 10.94272624  -0.2387107 ]
 [ 10.69160362   0.18757176]
 [ 10.493138     2.97533632]
 [ 10.493138     2.97533632]
 [ 10.38438779  -0.15065046]
 [  9.95119524  -2.68951514]
 [  9.95119524  -2.68951514]
 [  9.92970786  -2.4570964 ]
 [  9.95119524  -2.68951514]
 [  9.95119524  -2.68951514]
 [  9.94961505   2.29986791]
 [  9.73251403  -0.70330525]
 [  9.71102665  -0.47088652]
 [  9.79916187  -1.39525481]
 [  9.63770693  -0.31004687]
 [  8.60215655   3.85139575]
 [  9.40807168  -0.11618315]
 [  8.52032421  -2.52855173]
 [  8.62438014   1.42697707]
 [  8.24330761   1.86340753]
 [  8.06533586  -0.87958803]
 [  7.07189419   3.1352999 ]
 [  5.86746826   6.57715692]
 [  5.45109088   1.31705767]
 [  5.63478512   6.74343646]
 [  5.0278416    5.27116594]
 [  4.23201682   0.25583089]
 [  9.95809184   3.84944742]
 [  7.2088863    4.47558875]
 [  2.84602474   3.20756246]
 [-11.73217678   2.29381528]
 [-11.66102786  -0.24590344]
 [-11.260618     0.75588764]
 [-11.37667236   1.17658902]
 [-11.01061583   4.63737638]
 [-10.71966082   2.10028271]
 [-11.02751695   0.54373076]
 [-10.98586627   0.57698286]
 [-10.13092371   0.83636777]
 [-10.07517186   1.33131265]
 [ -9.9150317    2.58229108]
 [-10.27333074  -0.38447213]
 [-10.11028757   2.34561243]
 [-10.10682272  -1.51871386]
 [-10.53251967  -0.80475847]
 [-10.04344696   1.190844  ]
 [-10.11609831  -0.35795755]
 [-10.12721967  -0.118331  ]
 [-10.14034718  -2.24473056]
 [ -9.96166849  -1.07289562]
 [-10.15297121  -2.07833443]
 [ -9.65264669   0.83044796]
 [-10.1192188   -2.7538597 ]
 [ -9.64005546   0.41503953]
 [ -9.55694086   2.0255622 ]
 [ -9.49741134  -1.0743855 ]
 [ -9.48118265  -0.90229741]
 [ -9.38702931   1.5486255 ]
 [ -9.51451876  -2.56622035]
 [ -9.03623688   4.10921831]
 [ -9.08531027   0.30767929]
 [ -9.66781809  -2.27676154]
 [ -9.67195772  -2.48988254]
 [ -9.04974507   1.83882726]
 [ -9.42222402  -0.305662  ]
 [ -9.49637575  -0.52134363]
 [ -9.39943381  -0.41209569]
 [ -8.89124312   4.05621579]
 [ -9.10955393  -0.02814904]
 [ -9.44534663  -0.53299767]
 [ -9.47160831  -1.44264796]
 [ -8.75349155  -1.47702453]
 [ -8.75349155  -1.47702453]
 [ -9.28386981  -2.60223379]
 [ -8.26267577   2.18455125]
 [ -9.20723335  -2.90534591]
 [ -8.82933314  -0.08224272]
 [ -8.78382322   0.25779692]
 [ -9.3261147    0.47574662]
 [ -8.87952672   0.80265764]
 [ -9.0057564    0.06934773]
 [ -9.0102426   -1.9049662 ]
 [ -8.65457638   0.11842163]
 [ -9.17542043  -2.43265738]
 [ -9.17542043  -2.43265738]
 [ -8.74068158  -0.26845271]
 [ -8.83340794  -0.60389656]
 [ -8.54005128   2.32923657]
 [ -8.89491834   4.45183769]
 [ -8.80290914  -1.35445751]
 [ -8.69114341   0.32827108]
 [ -9.046022    -3.8459136 ]
 [ -8.56169639   0.59821103]
 [ -8.78087207  -3.66874861]
 [ -8.5604139   -1.26379765]
 [ -8.9092373   -4.62002304]
 [ -8.90644893  -3.83259997]
 [ -8.38769674   0.79899572]
 [ -8.68949986  -0.58588734]
 [ -8.44821052   1.8720181 ]
 [ -8.19472387   1.61459195]
 [ -8.30007113   1.79808861]
 [ -8.29402152   1.43808808]
 [ -8.3279063   -1.21106773]
 [ -8.11361932   1.20667667]
 [ -8.20866411  -0.63644964]
 [ -8.12051972   0.7641283 ]
 [ -8.03261425   3.31073116]
 [ -8.87033022  -4.03402834]
 [ -8.19226721   0.83524766]
 [ -8.30040794   2.40721665]
 [ -7.81397451   1.47764654]
 [ -8.23633581  -3.10968065]
 [ -8.12098504  -0.44136681]
 [ -8.14974944  -2.90298584]
 [ -8.31173534  -1.76697272]]
```

In [12]:
```python
#Classification phase
#Select necessary libraries
from sklearn.neighbors import KNeighborsClassifier
#Define KNN classifier model object
KNNClassifier_Object = KNeighborsClassifier(n_neighbors=5)
#Train KNN classifier model
Trained_KNNClassifier_Object=KNNClassifier_Object.fit(Compressed_XTrain_Sample, YTrain_Sample)
print(Trained_KNNClassifier_Object)
```

```
KNeighborsClassifier()
```

In [13]:
```python
from mlxtend.plotting import plot_decision_regions
from sklearn.metrics import confusion_matrix, plot_confusion_matrix
import matplotlib.pyplot as plt

#test it can classify new data and visualize the classifier
#use predict() function to predict classes of Test Sample
Predicted_YTest_Sample = Trained_KNNClassifier_Object.predict(Compressed_XTest_Sample)
 #visualize the testing results using Decision boundaries
from mlxtend.plotting import plot_decision_regions

plot_decision_regions(Compressed_XTest_Sample, Predicted_YTest_Sample, clf=Trained_KNNClassifier_Object,legend=2)
plt.title('Classification Test Results')
Legend=plt.legend()
Legend.get_texts()[0].set_text('Divorce_prediction')


plt.show()

#visualize Testing results using a Data Frame
Both_YTest_Sample={
    'Actual_Classes': Actual_YTest_Sample,
    'Predicted_Classes': Predicted_YTest_Sample
                   }
Tabulated_Test_Results = pd.DataFrame(Both_YTest_Sample, columns=['Actual_Classes','Predicted_Classes'])
print("Classification Test results\n",Tabulated_Test_Results)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\mlxtend\plotting\decision_regions.py:244: UserWarning: No contour levels w
ere found within the data range.
  ax.contour(xx, yy, Z, cset.levels,
```




```
Classification Test results
      Actual_Classes  Predicted_Classes
0                  1                  1
1                  1                  1
2                  1                  1
3                  1                  1
4                  1                  1
..               ...                ...
155                0                  1
156                0                  1
157                0                  1
158                0                  1
159                0                  1

[160 rows x 2 columns]
```

In [14]:
```python
#pip install pandas_ml
```

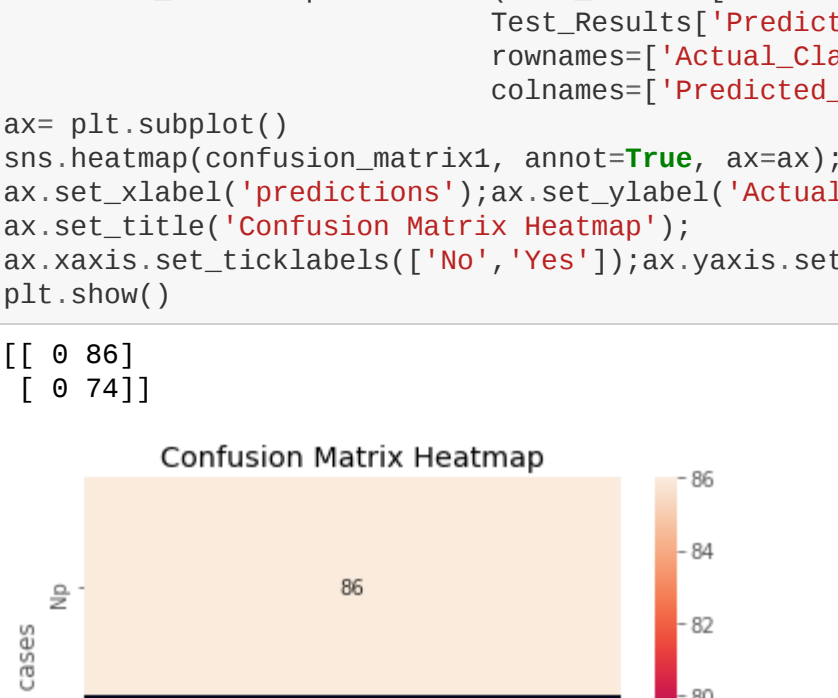
```
Requirement already satisfied: pandas_ml in c:\programdata\anaconda3\lib\site-packages (0.6.1)Note: you may need to r
estart the kernel to use updated packages.
Requirement already satisfied: enum34 in c:\programdata\anaconda3\lib\site-packages (from pandas_ml) (1.1.10)
Requirement already satisfied: pandas>=0.19.0 in c:\programdata\anaconda3\lib\site-packages (from pandas_ml) (1.0.5)
Requirement already satisfied: pytz>=2017.2 in c:\programdata\anaconda3\lib\site-packages (from pandas>=0.19.0->panda
s_ml) (2020.1)
Requirement already satisfied: python-dateutil>=2.6.1 in c:\programdata\anaconda3\lib\site-packages (from pandas>=0.1
9.0->pandas_ml) (2.8.1)
Requirement already satisfied: numpy>=1.13.3 in c:\programdata\anaconda3\lib\site-packages (from pandas>=0.19.0->pand
as_ml) (1.18.5)
Requirement already satisfied: six>=1.5 in c:\programdata\anaconda3\lib\site-packages (from python-dateutil>=2.6.1->p
andas>=0.19.0->pandas_ml) (1.15.0)
```

In [15]:
```python
#pip install jaccard_score
```

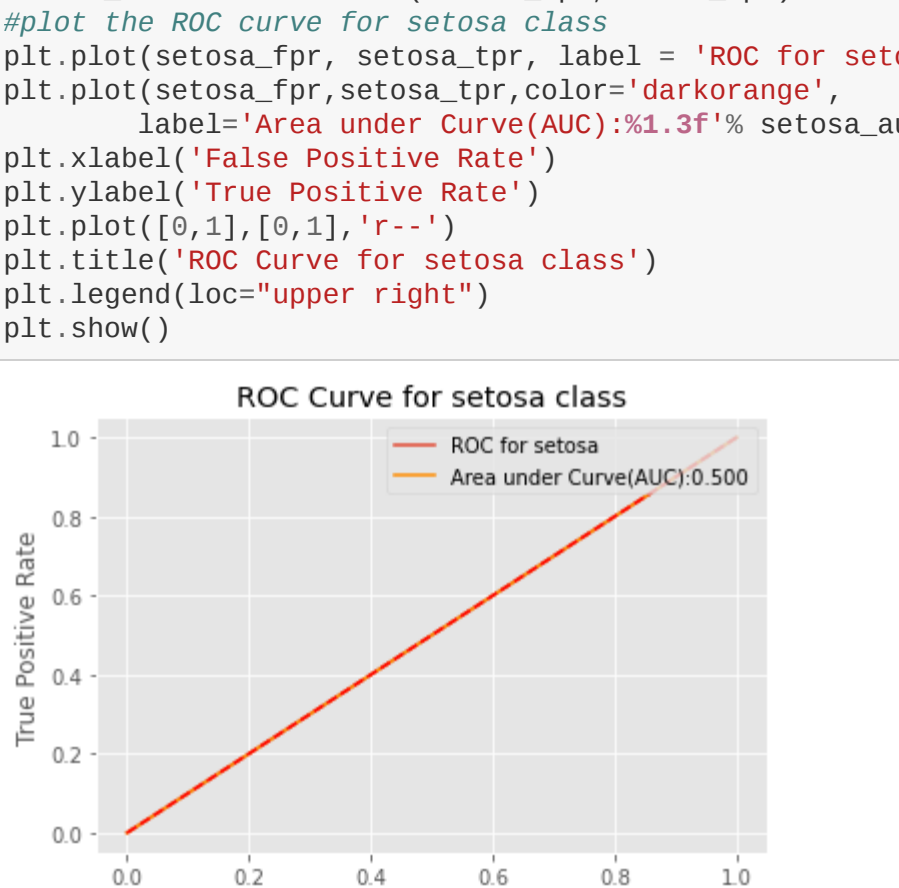
```
Note: you may need to restart the kernel to use updated packages.
ERROR: Could not find a version that satisfies the requirement jaccard_score (from versions: none)
ERROR: No matching distribution found for jaccard_score
```

In [19]:
```python
from sklearn.metrics import accuracy_score
#Test and validate the classifier
#validate the model to ensure its performance
from sklearn.metrics import confusion_matrix
Test_Results =pd.DataFrame(Both_YTest_Sample)
from sklearn.metrics import accuracy_score
Confusion_Matrix = confusion_matrix(Test_Results['Actual_Classes'], Test_Results['Predicted_Classes'])
print(Confusion_Matrix)

import seaborn as sns
#visualize the validation results using a heat map
confusion_matrix1=pd.crosstab(Test_Results['Actual_Classes'],
                              Test_Results['Predicted_Classes'],
                              rownames=['Actual_Classes'],
                              colnames=['Predicted_Classes'])
ax= plt.subplot()
sns.heatmap(confusion_matrix1, annot=True, ax=ax);
ax.set_xlabel('predictions');ax.set_ylabel('Actual cases');
ax.set_title('Confusion Matrix Heatmap');
ax.xaxis.set_ticklabels(['No','Yes']);ax.yaxis.set_ticklabels(['Np','Yes']);
plt.show()
```

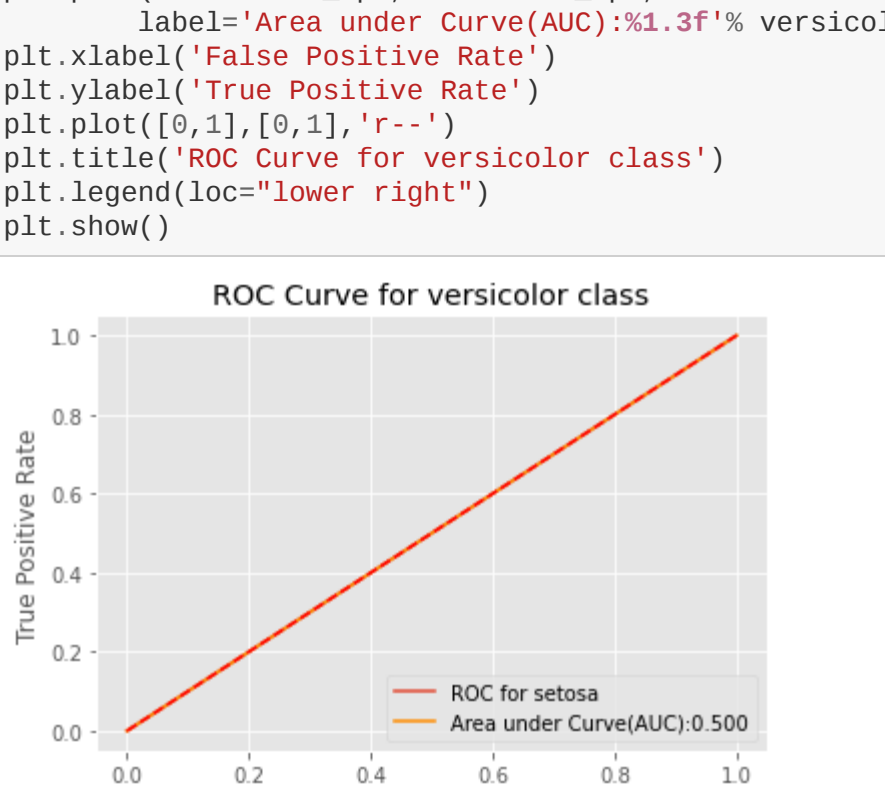
```
[[ 0 86]
 [ 0 74]]
```



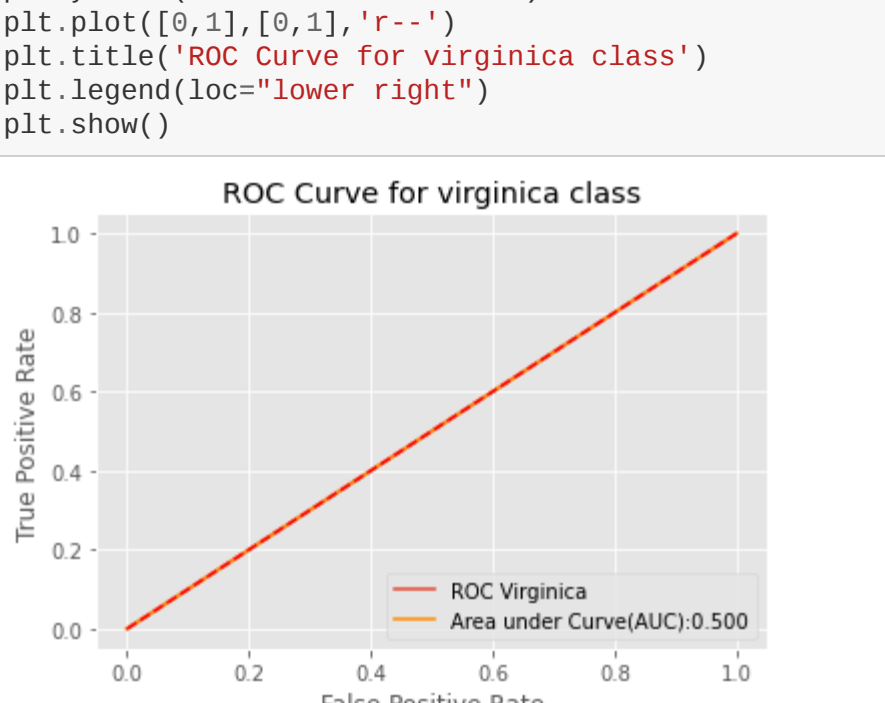
In [24]:
```python
import sklearn.metrics as metrics
#calculate the area under the ROC for setosa class
setosa_fpr, setosa_tpr,setosa_thresholds=metrics.roc_curve(Actual_YTest_Sample, Predicted_YTest_Sample,pos_label=0)
setosa_auc = metrics.auc(setosa_fpr,setosa_tpr)
#plot the ROC curve for setosa class
plt.plot(setosa_fpr, setosa_tpr, label = 'ROC for setosa')
plt.plot(setosa_fpr,setosa_tpr,color='darkorange',
        label='Area under Curve(AUC):%1.3f'% setosa_auc)
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.plot([0,1],[0,1],'r--')
plt.title('ROC Curve for setosa class')
plt.legend(loc="upper right")
plt.show()
```



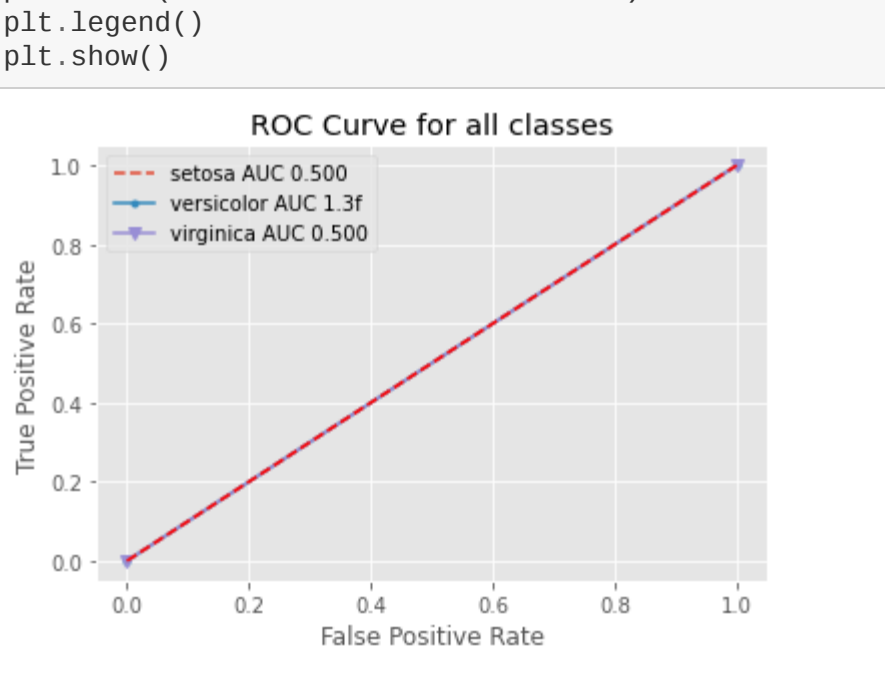
In [29]:
```python
#validation of the ROC Curve
#calculate are under the ROC curve for versicolor color
versicolor_fpr,versicolor_tpr,vesricolor_thresholds = metrics.roc_curve(Actual_YTest_Sample, Predicted_YTest_Sample,
pos_label=1)
versicolor_auc=metrics.auc(versicolor_fpr,versicolor_tpr)
#plot ROC curve

#plot the ROC curve for setosa class
plt.plot(versicolor_fpr, versicolor_tpr, label = 'ROC for setosa')
plt.plot(versicolor_fpr,versicolor_tpr,color='darkorange',
        label='Area under Curve(AUC):%1.3f'% versicolor_auc)
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.plot([0,1],[0,1],'r--')
plt.title('ROC Curve for versicolor class')
plt.legend(loc="lower right")
plt.show()
```



In [31]:
```python
#validation of the ROC Curve
#calculate are under the ROC curve for versicolor color
virginica_fpr,virginica_tpr,virginica_thresholds = metrics.roc_curve(Actual_YTest_Sample, Predicted_YTest_Sample,pos
_label=1)
virginica_auc=metrics.auc(virginica_fpr,virginica_tpr)
#plot the ROC curve for setosa class
plt.plot(virginica_fpr, virginica_tpr, label = 'ROC Virginica')
plt.plot(virginica_fpr,virginica_tpr,color='darkorange',
        label='Area under Curve(AUC):%1.3f'% virginica_auc)
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.plot([0,1],[0,1],'r--')
plt.title('ROC Curve for virginica class')
plt.legend(loc="lower right")
plt.show()
```



In [32]:
```python
#plot the roc curve for all the classes
plt.plot(setosa_fpr,setosa_tpr,linestyle='--',label='setosa AUC %1.3f' %setosa_auc)
plt.plot(versicolor_fpr,versicolor_tpr,marker='.', label='versicolor AUC 1.3f' %versicolor_auc)
plt.plot(virginica_fpr,virginica_tpr,marker='v',label='virginica AUC %1.3f' %virginica_auc)
#axis label
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.plot([0,1],[0,1],'r--')
plt.title('ROC Curve for all classes')
plt.legend()
plt.show()
```



In [ ]: