In [1]:
```python
#select necesary libraries
import numpy as np
from sklearn.metrics import accuracy_score
import pandas as pd
from sklearn import preprocessing
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import Normalizer
from sklearn.decomposition import PCA

#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 [2]:
```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 [3]:
```python
#select samples from the transformed target data
#selecting training and test sample
Training_Sample,Test_Sample=train_test_split(Numeric_Array,test_size=0.3,random_state=2)
print("Training Sample:\n", Training_Sample)
print("Test Sample:\n", Test_Sample)
#select input attributes from training sample
```

```
Training Sample:
 [[0 0 3 ... 0 0 0]
 [3 1 1 ... 0 4 0]
 [3 0 0 ... 1 0 0]
 ...
 [3 3 3 ... 3 3 1]
 [4 4 3 ... 4 4 1]
 [0 0 0 ... 3 1 0]]
Test Sample:
 [[3 4 3 ... 4 4 1]
 [3 3 3 ... 4 4 1]
 [3 4 3 ... 4 3 1]
 ...
 [0 0 0 ... 0 2 0]
 [3 3 2 ... 3 4 1]
 [2 0 2 ... 2 1 0]]
```

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

In [5]:
```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
 [0 0 0 0 0 0 0 1 1 1 1 0 0 0 1 1 0 1 0 1 0 0 0 1 0 1 1 0 0 0 0 0 0 1 1 1 0
 1 0 0 0 0 0 0 1 0 1 1 0 1 1 1 1 1 0 0 1 0 1 1 1 1 1 1 1 0 1 1 0 0 1 1 0 0
 0 1 1 0 0 1 1 0 0 1 1 1 0 0 0 1 0 1 1 0 1 1 1 0 1 0 1 0 1 0 0 0 1 1 1 0 0
 1 0 1 1 1 1 1 0]
```

In [6]:
```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

 [[3 4 3 ... 4 4 4]
 [3 3 3 ... 4 4 4]
 [3 4 3 ... 3 4 3]
 ...
 [0 0 0 ... 1 0 2]
 [3 3 2 ... 4 3 4]
 [2 0 2 ... 1 2 1]]
```

In [7]:
```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 0 1 1 0 0 1 0 1 1 0 1 0 1 1 1 0 0 0 0 1 1 0 0 1 0 1 0 1 0 0 0 1 0
 0 0 0 0 1 1 0 1 1 0 0 0 1 0]
```

In [8]:
```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:
 [[ -9.73915962   1.64680229]
 [ -8.93726983   0.12789023]
 [ -9.00522101   0.31561341]
 [ -9.11596152  -3.10562409]
 [ -9.73766554  -1.70898361]
 [ -9.23089836  -0.38960601]
 [ -9.08013017  -0.56888428]
 [ 14.82904093   0.53453589]
 [ 11.66648172  -0.17442586]
 [ 12.17234548  -1.01359425]
 [ 10.49710716   0.57362542]
 [ -9.25577424  -2.27164897]
 [-10.78941235  -1.37280351]
 [ -9.33714167  -1.30376121]
 [ -5.25600644  -1.31520221]
 [ 12.13752955  -1.02250238]
 [ -8.99560122  -0.78554075]
 [  5.23167028   1.10145482]
 [-11.1892132   -0.08702613]
 [ 12.90105612  -1.34448484]
 [ -9.09358265   0.02887838]
 [ -9.0404947    0.20893755]
 [ -8.5432773   -2.16360548]
 [ 12.17234548  -1.01359425]
 [ -9.38062186  -1.12820206]
 [ 13.41070976   2.08389658]
 [  9.69546174  -3.71278306]
 [ -8.62210063   2.43568409]
 [ -8.73548325   1.68176917]
 [ -9.60418522  -0.0776569 ]
 [-10.24490909   1.45913488]
 [ -8.88732274  -1.25833204]
 [-10.27082986   1.1206134 ]
 [ 12.90105612  -1.34448484]
 [ 10.47616709  -0.11434611]
 [ 13.49022317  -0.58351494]
 [ -8.33187566   0.08329763]
 [  7.82971052  -1.86389347]
 [ -9.22848552  -2.21060307]
 [ -9.70913472   0.2548897 ]
 [ -8.48954657  -3.38800529]
 [-11.60273933   0.79642974]
 [ -9.67783486  -0.9768093 ]
 [-11.92280108   2.22834782]
 [ 12.92199619  -0.65651331]
 [ -8.98238799  -1.16304768]
 [  9.65519363  -3.65358487]
 [  9.76854444   2.74066933]
 [ -9.71110425  -0.73181061]
 [  9.42064874  -0.65039072]
 [  5.9449173    3.91639317]
 [  7.09729787   5.68324305]
 [ 10.31721787   3.47258051]
 [ 13.41070976   2.08389658]
 [ -9.10063049  -1.54327848]
 [-11.4716459    1.05732628]
 [  9.19855751  -0.60362361]
 [-10.36437402  -0.55718083]
 [ -4.21194476   0.36470059]
 [  9.69546174  -3.71278306]
 [ 12.09283208   1.65381727]
 [ 10.89908768  -2.37209678]
 [ 14.82904093   0.53453589]
 [  5.65883531   3.21896897]
 [  1.05125365   6.11297595]
 [ -9.28862917   1.17123986]
 [ 12.17779765  -1.08170057]
 [ 12.13199979   0.79350036]
 [ -8.10561456  -3.05154524]
 [-10.31376017   1.94688865]
 [  2.58745439  -0.61657836]
 [  9.69546174  -3.71278306]
 [-11.22781488   1.33703909]
 [ -9.68397548  -0.35641657]
 [ -9.86454805   0.85590056]
 [ 10.31721787   3.47258051]
 [  8.04682169   1.45233907]
 [ -9.29592459  -0.10024824]
 [ -8.54450908  -1.81365585]
 [ 11.48272496   0.20487585]
 [ 10.93390361  -2.36318865]
 [ -8.31201894   1.06541536]
 [ -9.56251117   0.36089645]
 [ 11.01333943  -3.28270376]
 [ 10.79318203  -0.32455047]
 [  4.05731289   0.33000113]
 [ -8.57781905  -1.17344172]
 [ -9.1000003    1.86780482]
 [ -8.17572873   3.7888494 ]
 [ 10.93390361  -2.36318865]
 [-11.17318394   3.70768998]
 [  8.48206856   6.28354758]
 [  9.55474016  -1.67414488]
 [ -8.49578496   1.05063627]
 [ 10.16484727  -0.22520345]
 [ 12.03715367  -0.79095881]
 [ 11.40393228  -0.52248693]
 [-10.18198424  -0.18156532]
 [ 10.8389799   -2.1997514 ]
 [ -9.88524073  -1.68804592]
 [ -6.24265101   1.17700055]
 [-10.36889837  -1.04080745]
 [ 11.40393228  -0.52248693]
 [ -9.87953469  -0.8096858 ]
 [ -9.68796468  -0.34125188]
 [ -8.75848166  -0.70325954]
 [ 12.17226789   0.73430217]
 [  9.69546174  -3.71278306]
 [ 10.89363551  -2.30399046]
 [ -8.52647287   1.05177021]
 [-10.48947775  -0.57739814]
 [  2.69832461   5.19941512]
 [-10.34435887  -0.33981104]
 [  7.00334106   6.42549352]
 [ 10.93390361  -2.36318865]
 [ 12.64718319   2.40587904]
 [  8.42950117   1.53436159]
 [ 13.49022317  -0.58351494]
 [ -8.40945997  -2.97377109]]
```

In [9]:
```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:
 [[ 14.16629338  -0.05995038]
 [ 12.99382033  -0.75665428]
 [ 11.71918267   0.99597938]
 [ 12.99382033  -0.75665428]
 [ -9.58199849  -0.78641563]
 [ 14.16629338  -0.05995038]
 [ 12.32389914  -1.15294086]
 [ -8.19299882  -0.71591504]
 [-10.23450067   1.82988715]
 [ 11.75997199  -2.30624576]
 [ -8.80270584  -2.61302463]
 [  5.23464671   3.62721322]
 [  5.13483065   1.94414035]
 [ -7.89872074   0.48010672]
 [ 10.1910608   -0.87288274]
 [ -8.52849823   4.12597099]
 [ 11.75997199  -2.30624576]
 [  2.62376989   2.91416351]
 [ 12.93244504  -1.60954186]
 [ -7.65578024   1.52468389]
 [ -8.04934025   0.4535768 ]
 [ -8.67181543  -2.93447043]
 [ -8.40019397  -3.73916046]
 [ 11.37665701  -0.52202141]
 [ -5.62853808   0.99741447]
 [ -9.66348834  -1.87306591]
 [ -8.48603505  -4.44982959]
 [ 11.68144699  -1.36444005]
 [ -7.81959962   1.50938761]
 [ 13.09981437  -1.51367262]
 [ -8.52557524   4.27412802]
 [ 10.39616581   1.58340592]
 [ -9.17827565  -2.27925261]
 [ -8.27886214  -0.02448487]
 [ -7.74276906   0.69421357]
 [  6.20054778   7.28643739]
 [ -8.95279771   1.4134299 ]
 [ -9.42549258   2.63434722]
 [ -9.59760903  -2.82627282]
 [ -7.7279913   -0.57628747]
 [ -8.67181543  -2.93447043]
 [  5.54893173   6.61077088]
 [ 14.01709153   0.02216284]
 [-11.19719254  -0.44111276]
 [ 11.75997199  -2.30624576]
 [  9.0884359   -1.9529604 ]
 [ -9.65624072   0.40065463]
 [ -8.94538002   0.12672066]
 [ -8.23304253  -1.53242044]
 [ 10.21852984  -0.68809536]
 [ -7.64034151   0.50588985]]
```

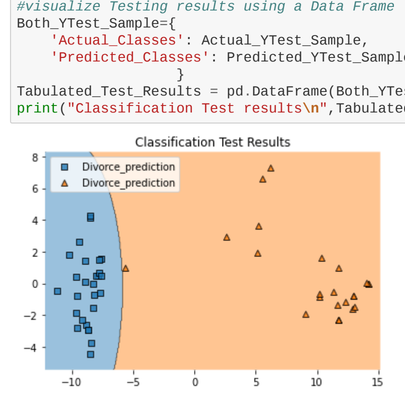
In [10]:
```python
#Classification phase
#develop a NaiveBayes Classifier
from sklearn.naive_bayes import GaussianNB
Classifier_object = GaussianNB()
#train the classifier object to develop a classifier
NBayes_classifier=Classifier_object.fit(Compressed_XTrain_Sample,YTrain_Sample)
```

In [11]:
```python
from mlxtend.plotting import plot_decision_regions
from sklearn.metrics import confusion_matrix, plot_confusion_matrix
import matplotlib.pyplot as plt
```

In [ ]:

In [12]:
```python
#test it can classify new data and visualize the classifier
#use predict() function to predict classes of Test Sample
Predicted_YTest_Sample = NBayes_classifier.predict(Compressed_XTest_Sample)
 #visualize the testing results using Decision boundaries
plot_decision_regions(Compressed_XTest_Sample, Predicted_YTest_Sample, clf=Classifier_object,legend=2)
plt.title('Classification Test Results')
Legend=plt.legend()
Legend.get_texts()[0].set_text('Divorce_prediction')
Legend.get_texts()[1].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)
```



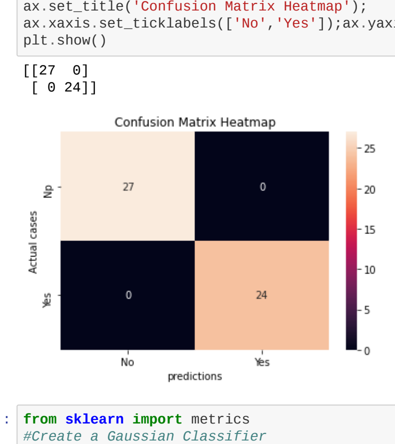

```
Classification Test results
     Actual_Classes  Predicted_Classes
0                 1                  1
1                 1                  1
2                 1                  1
3                 1                  1
4                 0                  0
5                 1                  1
6                 1                  1
7                 0                  0
8                 0                  0
9                 1                  1
10                0                  0
11                1                  1
12                1                  1
13                0                  0
14                1                  1
15                0                  0
16                1                  1
17                1                  1
18                1                  1
19                0                  0
20                0                  0
21                0                  0
22                0                  0
23                1                  1
24                1                  1
25                0                  0
26                0                  0
27                1                  1
28                0                  0
29                1                  1
30                0                  0
31                1                  1
32                0                  0
33                0                  0
34                0                  0
35                1                  1
36                0                  0
37                0                  0
38                0                  0
39                0                  0
40                0                  0
41                1                  1
42                1                  1
43                0                  0
44                1                  1
45                1                  1
46                0                  0
47                0                  0
48                0                  0
49                1                  1
50                0                  0
```

In [ ]:

In [16]:
```python
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()
```

```
[[27  0]
 [ 0 24]]
```



In [17]:
```python
from sklearn import metrics
#Create a Gaussian Classifier
gnb = GaussianNB()
gnb.fit(XTrain_Sample, YTrain_Sample)
y_pred = gnb.predict(XTest_Sample)
# Model Accuracy, how often is the classifier correct?
accuracy = metrics.accuracy_score(Actual_YTest_Sample, y_pred)
print("Accuracy: %.2f" % (accuracy*100))
```

```
Accuracy: 98.04
```

In [ ]: