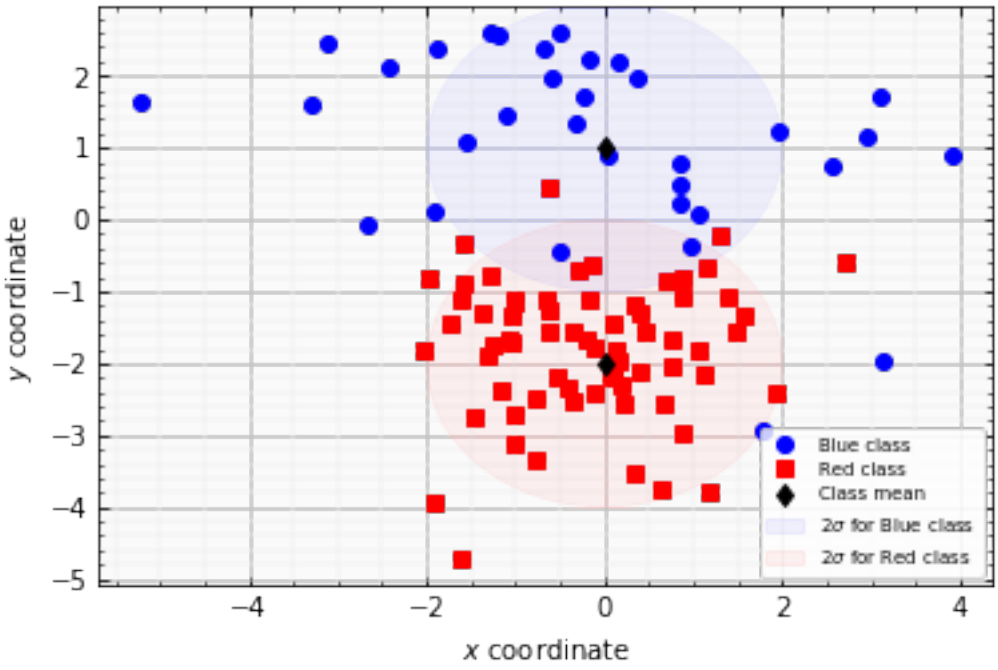
Mid

October 9, 2020

[30]:
```
import numpy as np
import matplotlib
from matplotlib import pyplot as plt
from matplotlib.colors import LinearSegmentedColormap
from sklearn import *
import math
from sklearn import neighbors
from sklearn.naive_bayes import GaussianNB
```

[31]:
```
BlueDataSet = np.random.multivariate_normal([0,1],[[4,0],[0,2]],33) #label 0
RedDataSet = np.random.multivariate_normal([0,-2],[[1,0],[0,1]],67) #label 1

TrainingData = np.r_[RedDataSet,BlueDataSet]
TrainingLabel = np.r_[np.ones((67,1)),0*np.ones((33,1))]

TestData_b = np.random.multivariate_normal([0,1],[[4,0],[0,2]],5000)
TestData_r = np.random.multivariate_normal([0,-2],[[1,0],[0,1]],5000)
TestData = np.r_[TestData_r,TestData_b]
TestLabel = np.r_[np.ones((5000,1)),0*np.ones((5000,1))]

fid = plt.figure()

Axes = plt.subplot(1,1,1)
Axes.axes.tick_params(which='both',direction='in',top=True, right=True)
plt.minorticks_on()
Axes.set_facecolor((0,0,0,0.02))
plt.plot(BlueDataSet[:,0],BlueDataSet[:,1],'o',color='b',label='Blue class')
plt.plot(RedDataSet[:,0],RedDataSet[:,1],'s',color='r',label='Red class')
plt.plot(0,1,'kd',label='Class mean')
plt.plot(0,-2,'kd')
BlueClassSdev = matplotlib.patches.Circle((0,1),radius=2,color='blue',alpha=0.
↪05,label='2$\sigma$ for Blue class')
Axes.add_patch(BlueClassSdev)
RedClassSdev = matplotlib.patches.Circle((0,-2),radius=2,color='red',alpha=0.
↪05,label='2$\sigma$ for Red class')
Axes.add_patch(RedClassSdev)
plt.grid(True,which='major',linewidth=0.5)
```

```
plt.grid(True,which='minor',linewidth=0.1)
# plt.xlim([-5,5])
# plt.ylim([-4,4])
plt.xlabel("$x$ coordinate")
plt.ylabel("$y$ coordinate")
plt.legend(loc='lower right',fontsize='x-small')
plt.savefig('Q1.jpg')
```



[32]:
```
my_colors = [(0,0,1),(1,0,0)]#RGB
my_cm = LinearSegmentedColormap.from_list('my_cm', my_colors, N=2)
```

1 Bayes

[33]:
```
#Need to create a mesh to evaluate classifier
x_grid = np.linspace(1.0*np.min(TrainingData[:,0]), 1.5*np.max(TrainingData[:
 ↪,0]), 100)
y_grid = np.linspace(1.0*np.min(TrainingData[:,1]), 1.5*np.max(TrainingData[:
 ↪,1]), 100)
x_mesh, y_mesh = np.meshgrid(x_grid, y_grid)
```

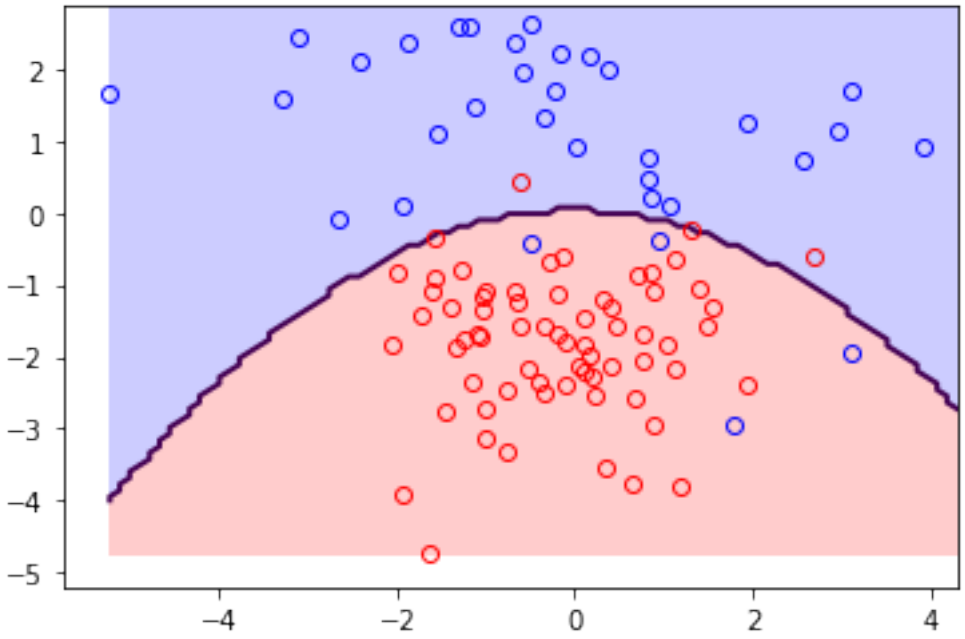
[34]:
```
def BayesPredict(xcoord, ycoord):
    pred = np.zeros(xcoord.shape)
    for i in np.arange(0, len(xcoord)):
```

```
        lin_class = 3*xcoord[i]**2+20*ycoord[i]+14-20*np.log(2)
        if lin_class >= 0:
            pred[i] = 0
        else:
            pred[i] = 1
    return (pred)

Test_prediction = BayesPredict(TestData[:,0], TestData[:,1])
count=0
for i in range(len(Test_prediction)):
    if Test_prediction[i] != TestLabel[i]:
        count+=1
print("test risk=", count/10000)

prediction = BayesPredict(x_mesh.ravel(), y_mesh.ravel())
prediction = prediction.reshape(x_mesh.shape)
```

```
test risk= 0.0973
```

[35]:
```
plt.plot(BlueDataSet[:,0], BlueDataSet[:,1], 'o', markerfacecolor="None",␣
 ↪markeredgecolor='b', label='Blue')
plt.plot(RedDataSet[:,0], RedDataSet[:,1], 'o', markerfacecolor="None",␣
 ↪markeredgecolor='r', label='Red')
plt.xlim([1.1*np.min(TrainingData[:,0]), 1.1*np.max(TrainingData[:,0])])
plt.ylim([1.1*np.min(TrainingData[:,1]), 1.1*np.max(TrainingData[:,1])])
Bayes = plt.contourf(x_mesh, y_mesh, prediction, levels = [-1,0,1], cmap=my_cm,␣
 ↪alpha=0.2)
Bayes = plt.contour(x_mesh, y_mesh, prediction, [-1,0,1,2], linewidths=2,␣
 ↪color='k')
plt.savefig('Q2.jpg')
```

```
/Users/gexueren/opt/anaconda3/lib/python3.7/site-
packages/ipykernel_launcher.py:6: UserWarning: No contour levels were found
within the data range.

/Users/gexueren/opt/anaconda3/lib/python3.7/site-
packages/ipykernel_launcher.py:6: UserWarning: The following kwargs were not
used by contour: 'color'
```



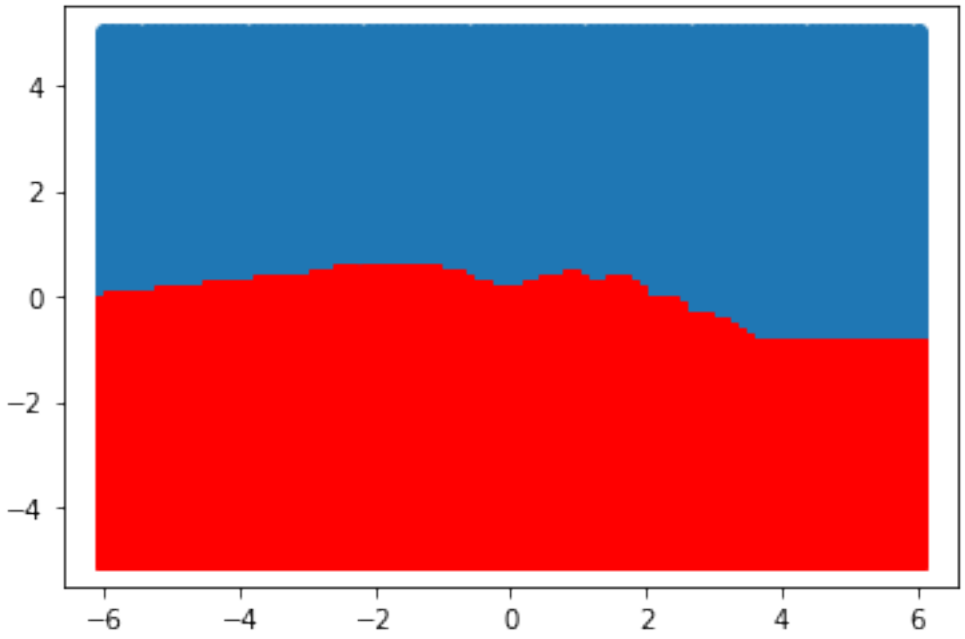
2 KNN

[36]:
```
NearestNeighbors = 15
KNNClassifier = neighbors.KNeighborsClassifier(n_neighbors=NearestNeighbors)
# TrainingData = np.r_[RedDataSet,BlueDataSet]
# TrainingLabel = np.r_[np.ones((67,1)),0*np.ones((33,1))]
KNNClassifier.fit(TrainingData,TrainingLabel.ravel())
Test_prediction = KNNClassifier.predict(TestData)
count=0
for i in range(len(Test_prediction)):
    if Test_prediction[i] != TestLabel[i]:
        count+=1
print("test risk=", count/10000)

x_grid = np.linspace(-6,6,100)
y_grid = np.linspace(-5,5,100)
x_mesh, y_mesh = np.meshgrid(x_grid,y_grid)
PredictedLabel = KNNClassifier.predict(np.c_[x_mesh.ravel(),y_mesh.ravel()])
Xcoord = x_mesh.ravel()
Ycoord = y_mesh.ravel()
BlueRegionX = Xcoord[PredictedLabel==0]
BlueRegionY = Ycoord[PredictedLabel==0]
RedRegionX = Xcoord[PredictedLabel==1]
RedRegionY = Ycoord[PredictedLabel==1]
plt.plot(BlueRegionX,BlueRegionY.ravel(),'o')
```

```
plt.plot(RedRegionX,RedRegionY.ravel(),'rs')
```

```
test risk= 0.1479
```

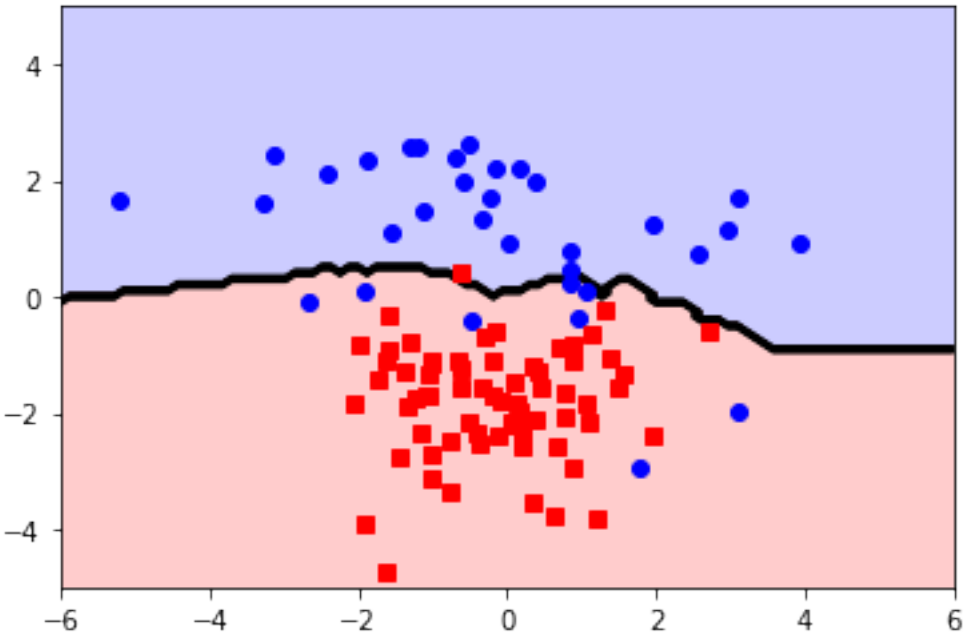
[36]:
```
[<matplotlib.lines.Line2D at 0x1a1ce27e90>]
```



[37]:
```
my_colors = [(0,0,1),(1,0,0)]#RGB
my_cm = LinearSegmentedColormap.from_list('my_cm', my_colors, N=2)
```

[38]:
```
plt.contourf(x_mesh,y_mesh,PredictedLabel.reshape(100,100),levels=[-1,0,1],␣
 ↪cmap = my_cm, alpha=0.2)
# plt.contourf(x_mesh,y_mesh,PredictedLabel.reshape(100,100),cmap = my_cm,␣
 ↪alpha=0.2)
plt.plot(BlueDataSet[:,0],BlueDataSet[:,1],'o',color='b',label='Blue class')
plt.plot(RedDataSet[:,0],RedDataSet[:,1],'s',color='r',label='Red class')
plt.contour(x_mesh,y_mesh,PredictedLabel.
 ↪reshape(100,100),linewidth=2,colors='k')
plt.savefig('Q3.jpg')
```

```
/Users/gexueren/opt/anaconda3/lib/python3.7/site-
packages/ipykernel_launcher.py:5: UserWarning: The following kwargs were not
used by contour: 'linewidth'
  """
```



3 LDA classifier

[39]:
```
#Need to create a mesh to evaluate classifier
x_grid = np.linspace(1.0*np.min(TrainingData[:,0]), 1.5*np.max(TrainingData[:
 ↪,0]), 100)
y_grid = np.linspace(1.0*np.min(TrainingData[:,1]), 1.5*np.max(TrainingData[:
 ↪,1]), 100)
x_mesh, y_mesh = np.meshgrid(x_grid, y_grid)
```

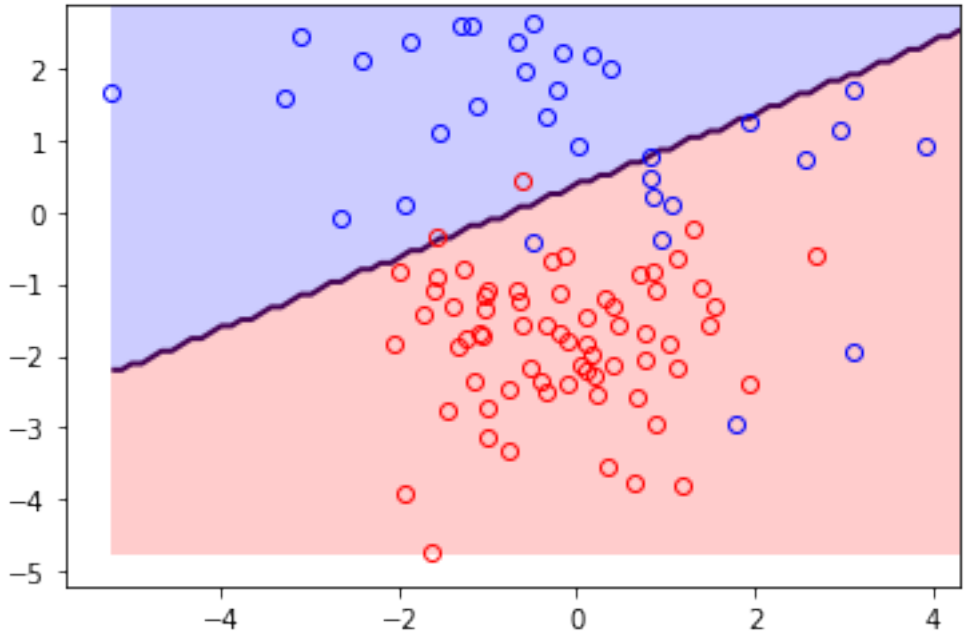
[40]:
```
mu_b = np.transpose([[-2,1]])
mu_r = np.transpose([[2,-1]])

pi_b = 1/3
pi_r = 2/3

sigma_1 =[[0,0],[0,0]]
for i in range(len(BlueDataSet)):
    sigma_1 += (np.transpose([[BlueDataSet[i][0],BlueDataSet[i][1]]]) - mu_b).
 ↪dot(np.transpose(np.transpose([[BlueDataSet[i][0],BlueDataSet[i][1]]]) -␣
 ↪mu_b))
sigma_2 =[[0,0],[0,0]]
for i in range(len(BlueDataSet)):
    sigma_2 += (np.transpose([[RedDataSet[i][0],RedDataSet[i][1]]]) - mu_r).
 ↪dot(np.transpose(np.transpose([[RedDataSet[i][0],RedDataSet[i][1]]]) - mu_r))
```

```
Sigma = (sigma_1 + sigma_2) / 100
invSigma = np.linalg.inv(Sigma)

def LDAPredict(xcoord, ycoord):
    pred = np.zeros(xcoord.shape)
    for i in np.arange(0, len(xcoord)):
        lin_class_b = 1/2*((np.array([[xcoord[i]],[ycoord[i]]]) - mu_b).T).
 ↪dot(invSigma).dot(np.array([[xcoord[i]],[ycoord[i]]]) - mu_b) - np.log(pi_b)
        lin_class_r = 1/2*((np.array([[xcoord[i]],[ycoord[i]]]) - mu_r).T).
 ↪dot(invSigma).dot(np.array([[xcoord[i]],[ycoord[i]]]) - mu_r) - np.log(pi_r)
        if lin_class_b > lin_class_r:
            pred[i] = 1
        else:
            pred[i] = 0
    return (pred)

Test_prediction = LDAPredict(TestData[:,0], TestData[:,1])
count=0
for i in range(len(Test_prediction)):
    if Test_prediction[i] != TestLabel[i]:
        count+=1
print("test risk=", count/10000)

prediction = LDAPredict(x_mesh.ravel(), y_mesh.ravel())
prediction = prediction.reshape(x_mesh.shape)
```

```
test risk= 0.1843
```

[41]:
```
plt.plot(BlueDataSet[:,0], BlueDataSet[:,1], 'o', markerfacecolor="None",␣
 ↪markeredgecolor='b', label='Blue')
plt.plot(RedDataSet[:,0], RedDataSet[:,1], 'o', markerfacecolor="None",␣
 ↪markeredgecolor='r', label='Red')
plt.xlim([1.1*np.min(TrainingData[:,0]), 1.1*np.max(TrainingData[:,0])])
plt.ylim([1.1*np.min(TrainingData[:,1]), 1.1*np.max(TrainingData[:,1])])
Bayes = plt.contourf(x_mesh, y_mesh, prediction, levels = [-1,0,1], cmap=my_cm,␣
 ↪alpha=0.2)
Bayes = plt.contour(x_mesh, y_mesh, prediction, [-1,0,1,2], linewidths=2,␣
 ↪color='k')
plt.savefig('Q4.jpg')
```

```
/Users/gexueren/opt/anaconda3/lib/python3.7/site-
packages/ipykernel_launcher.py:6: UserWarning: No contour levels were found
within the data range.

/Users/gexueren/opt/anaconda3/lib/python3.7/site-
packages/ipykernel_launcher.py:6: UserWarning: The following kwargs were not
used by contour: 'color'
```



4 QDA

[42]:
```
#Need to create a mesh to evaluate classifier
x_grid = np.linspace(1.0*np.min(TrainingData[:,0]), 1.5*np.max(TrainingData[:
 ↪,0]), 100)
y_grid = np.linspace(1.0*np.min(TrainingData[:,1]), 1.5*np.max(TrainingData[:
 ↪,1]), 100)
x_mesh, y_mesh = np.meshgrid(x_grid, y_grid)
```

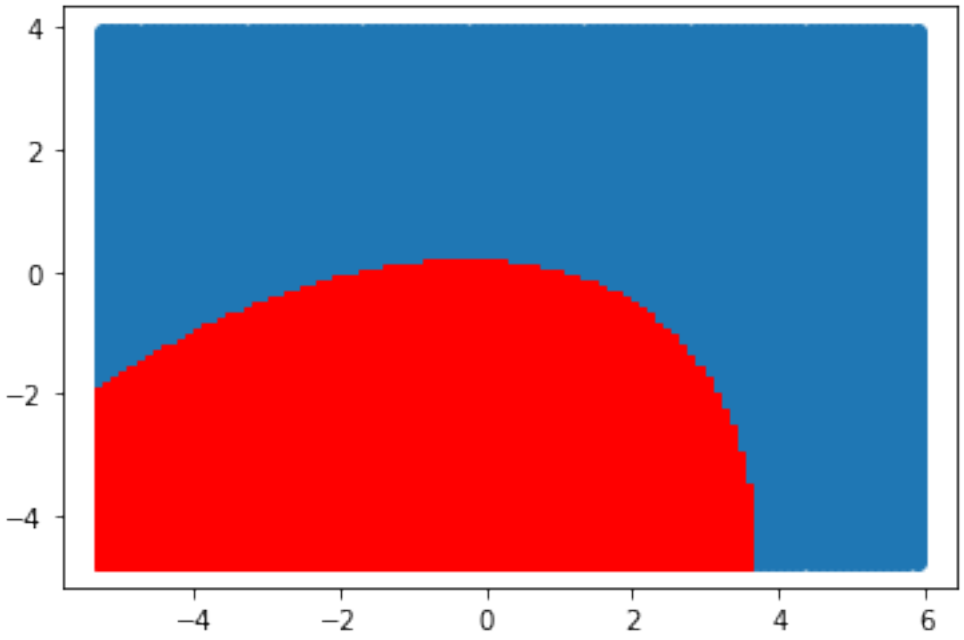
[43]:
```
QDAmodel = discriminant_analysis.QuadraticDiscriminantAnalysis()
QDAmodel.fit(TrainingData, TrainingLabel)

Test_prediction = QDAmodel.predict(TestData)
count=0
for i in range(len(Test_prediction)):
    if Test_prediction[i] != TestLabel[i]:
        count+=1
print("test risk=", count/10000)

PredictedLabel = QDAmodel.predict(np.c_[x_mesh.ravel(),y_mesh.ravel()])
Xcoord = x_mesh.ravel()
Ycoord = y_mesh.ravel()
BlueRegionX = Xcoord[PredictedLabel==0]
```

```
BlueRegionY = Ycoord[PredictedLabel==0]
RedRegionX = Xcoord[PredictedLabel==1]
RedRegionY = Ycoord[PredictedLabel==1]
plt.plot(BlueRegionX,BlueRegionY.ravel(),'o')
plt.plot(RedRegionX,RedRegionY.ravel(),'rs')
```

```
test risk= 0.1078
```

```
/Users/gexueren/opt/anaconda3/lib/python3.7/site-
packages/sklearn/utils/validation.py:760: DataConversionWarning: A column-vector
y was passed when a 1d array was expected. Please change the shape of y to
(n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
```

[43]:
```
[<matplotlib.lines.Line2D at 0x1a1c3eb090>]
```

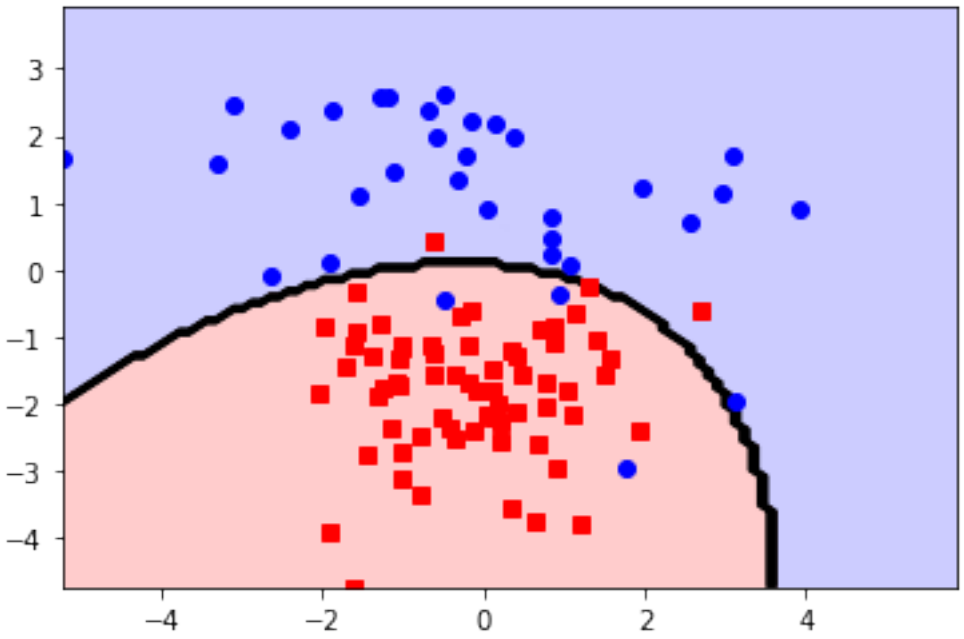


[44]:
```
my_colors = [(0,0,1),(1,0,0)]#RGB
my_cm = LinearSegmentedColormap.from_list('my_cm', my_colors, N=2)
```

[45]:
```
plt.contourf(x_mesh,y_mesh,PredictedLabel.reshape(100,100),levels=[-1,0,1],␣
↪cmap = my_cm, alpha=0.2)
# plt.contourf(x_mesh,y_mesh,PredictedLabel.reshape(100,100),cmap = my_cm,␣
↪alpha=0.2)
plt.plot(BlueDataSet[:,0],BlueDataSet[:,1],'o',color='b',label='Blue class')
plt.plot(RedDataSet[:,0],RedDataSet[:,1],'s',color='r',label='Red class')
```

```
plt.contour(x_mesh,y_mesh,PredictedLabel.
 ↪reshape(100,100),linewidth=2,colors='k')
plt.savefig('Q5.jpg')
```

```
/Users/gexueren/opt/anaconda3/lib/python3.7/site-
packages/ipykernel_launcher.py:5: UserWarning: The following kwargs were not
used by contour: 'linewidth'
  """
```



5 Logistic Regression

[46]:
```
#Need to create a mesh to evaluate classifier
x_grid = np.linspace(1.0*np.min(TrainingData[:,0]), 1.5*np.max(TrainingData[:
 ↪,0]), 100)
y_grid = np.linspace(1.0*np.min(TrainingData[:,1]), 1.5*np.max(TrainingData[:
 ↪,1]), 100)
x_mesh, y_mesh = np.meshgrid(x_grid, y_grid)
```

[47]:
```
mu_b = np.transpose([[0,1]])
mu_r = np.transpose([[0,-2]])

pi0 = 1/3
pi1 = 2/3
```

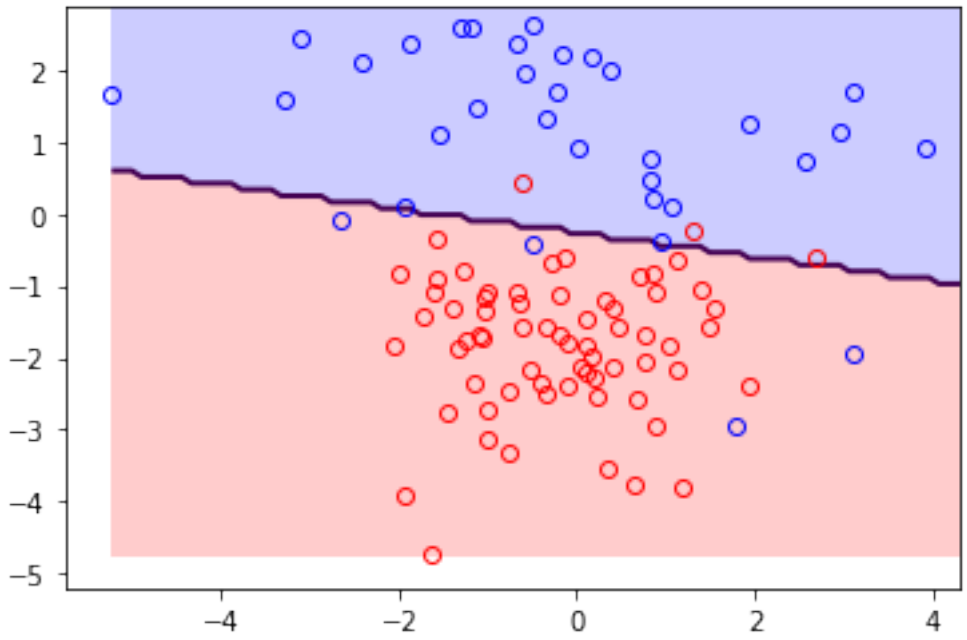
```
sigma_1 =[[0,0],[0,0]]
for i in range(len(BlueDataSet)):
    sigma_1 += (np.transpose([[BlueDataSet[i][0],BlueDataSet[i][1]]]) - mu_b).
 ↪dot(np.transpose(np.transpose([[BlueDataSet[i][0],BlueDataSet[i][1]]]) -␣
 ↪mu_b))
sigma_2 =[[0,0],[0,0]]
for i in range(len(BlueDataSet)):
    sigma_2 += (np.transpose([[RedDataSet[i][0],RedDataSet[i][1]]]) - mu_r).
 ↪dot(np.transpose(np.transpose([[RedDataSet[i][0],RedDataSet[i][1]]]) - mu_r))
Sigma = (sigma_1 + sigma_2)/100

invSigma = np.linalg.inv(Sigma)

w = invSigma.dot(mu_r - mu_b)
b = 0.5*(np.transpose(mu_b).dot(invSigma.dot(mu_b)) - np.transpose(mu_r).
 ↪dot(invSigma.dot(mu_r)))+np.log(pi1/pi0)

def LogisticPredict(xcoord, ycoord):
    pred = np.zeros(xcoord.shape)
    for i in np.arange(0, len(xcoord)):
        lin_class = np.transpose(w).dot([[xcoord[i]], [ycoord[i]]]) + b
        if lin_class >= 0:
            pred[i] = 1
        else:
            pred[i] = 0
    return (pred)

Test_prediction = LogisticPredict(TestData[:,0], TestData[:,1])
count=0
for i in range(len(Test_prediction)):
    if Test_prediction[i] != TestLabel[i]:
        count+=1
print("test risk=", count/10000)

prediction = LogisticPredict(x_mesh.ravel(), y_mesh.ravel())
prediction = prediction.reshape(x_mesh.shape)
```

```
test risk= 0.1182
```

[48]:
```
plt.plot(BlueDataSet[:,0], BlueDataSet[:,1], 'o', markerfacecolor="None",␣
 ↪markeredgecolor='b', label='Blue')
plt.plot(RedDataSet[:,0], RedDataSet[:,1], 'o', markerfacecolor="None",␣
 ↪markeredgecolor='r', label='Red')
plt.xlim([1.1*np.min(TrainingData[:,0]), 1.1*np.max(TrainingData[:,0])])
plt.ylim([1.1*np.min(TrainingData[:,1]), 1.1*np.max(TrainingData[:,1])])
Bayes = plt.contourf(x_mesh, y_mesh, prediction, levels = [-1,0,1], cmap=my_cm,␣
 ↪alpha=0.2)
```

```
Bayes = plt.contour(x_mesh, y_mesh, prediction, [-1,0,1,2], linewidths=2,␣
↪color='k')
plt.savefig('Q6.jpg')
```

```
/Users/gexueren/opt/anaconda3/lib/python3.7/site-
packages/ipykernel_launcher.py:6: UserWarning: No contour levels were found
within the data range.

/Users/gexueren/opt/anaconda3/lib/python3.7/site-
packages/ipykernel_launcher.py:6: UserWarning: The following kwargs were not
used by contour: 'color'
```



6 Gaussian Naive Bayes classifier

[49]:
```
GNBmodel = GaussianNB()
GNBmodel.fit(TrainingData,TrainingLabel)
```

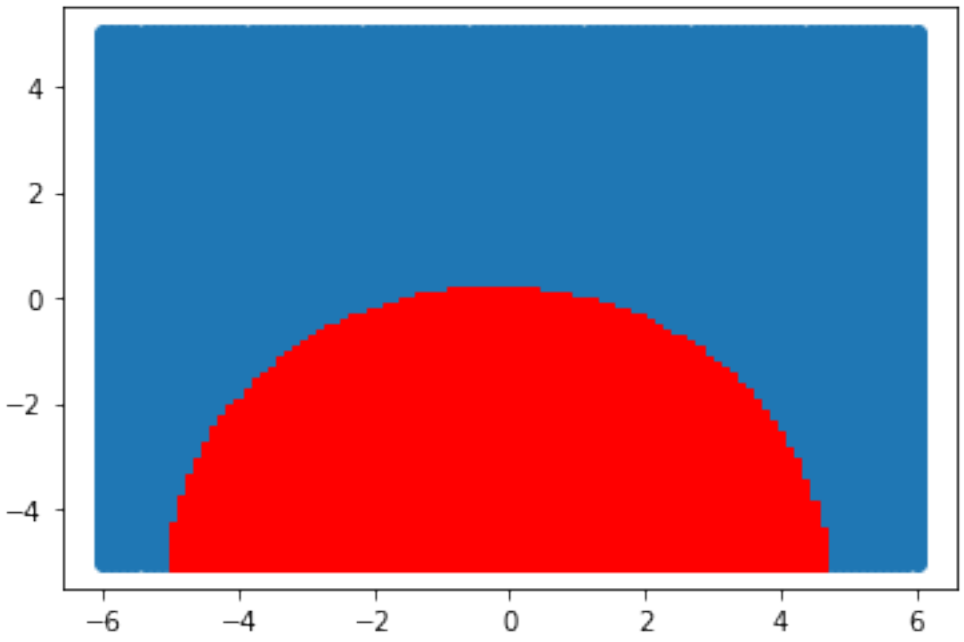
```
/Users/gexueren/opt/anaconda3/lib/python3.7/site-
packages/sklearn/naive_bayes.py:206: DataConversionWarning: A column-vector y
was passed when a 1d array was expected. Please change the shape of y to
(n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
```

[49]:
```
GaussianNB(priors=None, var_smoothing=1e-09)
```

[50]:
```
x_grid = np.linspace(-6,6,100)
y_grid = np.linspace(-5,5,100)
x_mesh, y_mesh = np.meshgrid(x_grid,y_grid)
PredictedLabel = GNBmodel.predict(np.c_[x_mesh.ravel(),y_mesh.ravel()])

Test_prediction = GNBmodel.predict(TestData)
count=0
for i in range(len(Test_prediction)):
    if Test_prediction[i] != TestLabel[i]:
        count+=1
print("test risk=", count/10000)

Xcoord = x_mesh.ravel()
Ycoord = y_mesh.ravel()
BlueRegionX = Xcoord[PredictedLabel==0]
BlueRegionY = Ycoord[PredictedLabel==0]
RedRegionX = Xcoord[PredictedLabel==1]
RedRegionY = Ycoord[PredictedLabel==1]
plt.plot(BlueRegionX,BlueRegionY.ravel(),'o')
plt.plot(RedRegionX,RedRegionY.ravel(),'rs')
```

```
test risk= 0.1052
```

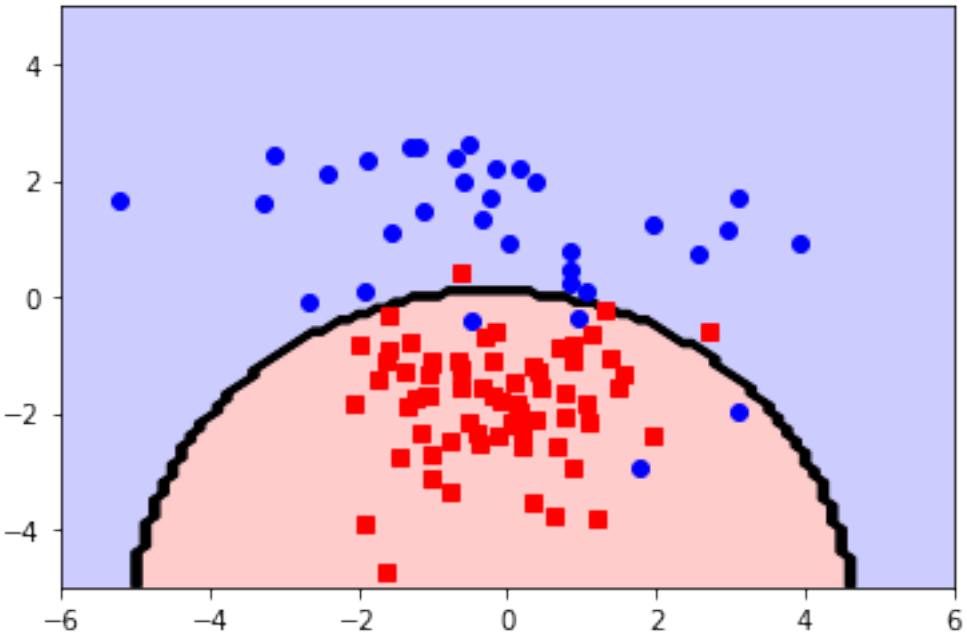
[50]:
```
[<matplotlib.lines.Line2D at 0x1a1c1ce350>]
```



[51]:
```
my_colors = [(0,0,1),(1,0,0)]#RGB
my_cm = LinearSegmentedColormap.from_list('my_cm', my_colors, N=2)
```

[52]:
```
plt.contourf(x_mesh,y_mesh,PredictedLabel.reshape(100,100),levels=[-1,0,1],␣
 ↪cmap = my_cm, alpha=0.2)
# plt.contourf(x_mesh,y_mesh,PredictedLabel.reshape(100,100),cmap = my_cm,␣
 ↪alpha=0.2)
plt.plot(BlueDataSet[:,0],BlueDataSet[:,1],'o',color='b',label='Blue class')
plt.plot(RedDataSet[:,0],RedDataSet[:,1],'s',color='r',label='Red class')
plt.contour(x_mesh,y_mesh,PredictedLabel.
 ↪reshape(100,100),linewidth=2,colors='k')
plt.savefig('Q7.jpg')
```

```
/Users/gexueren/opt/anaconda3/lib/python3.7/site-
packages/ipykernel_launcher.py:5: UserWarning: The following kwargs were not
used by contour: 'linewidth'
  """
```



[ ]: