Si Wu - Supervised ML

Homework 4

1) likelihood $= L = \underbrace{\prod_{i=1}^{N}}_{\text{independent}} \pi_i^{y_i} (1-\pi_i)^{1-y_i}$ → with individual observations

$$\log L = \sum_{i=1}^{N} \left[y_i \log \pi_i + (1-y_i) \log (1-\pi_i) \right]$$

NLL (negative log-likelihood)

$$= -\log L = \sum_{i=1}^{N} \left[-y_i \log \pi_i - (1-y_i) \log (1-\pi_i) \right]$$

$$\pi_i = \frac{1}{1+e^{-(\theta_0+\theta_1 x_i)}}$$

So NLL for one observation

$$= -y \log \frac{1}{1+e^{-(\theta_0+\theta_1 x)}} - (1-y) \log \left(1 - \frac{1}{1+e^{-(\theta_0+\theta_1 x)}}\right)$$

It helps to use chain rule

Define $\pi(z) = \frac{1}{1+e^{-z}}$. $z = \theta_0 + \theta_1 x$.

so, NLL $= -y [\log(\pi(z)) - (1-y) \log (1-\pi(z))$

$$\frac{d\pi}{dz} = \frac{d[(1+e^{-z})^{-1}]}{dz} = -(1+e^{-z})^{-2} \cdot e^{-z} \cdot (-1)$$

$$= \frac{e^{-z}}{(1+e^{-z})^2}$$

$$= \frac{1}{1+e^{-z}} \cdot \frac{e^{-z}}{1+e^{-z}} = \pi(z)\left(\frac{1+e^{-z}-1}{1+e^{-z}}\right)$$

$$= \pi(z)\left(1-\frac{1}{1+e^{-z}}\right)$$

$$= \pi(z)\left[1-\pi(z)\right].$$

So, $\frac{d(NLL)}{d\theta_1} = \frac{d(NLL)}{d\pi} \cdot \frac{d\pi}{dz} \cdot \frac{dz}{d\theta_1}$

$$= \frac{d\left[-Y\log\pi - (1-Y)\log(1-\pi)\right]}{d\pi} \cdot \pi(z)(1-\pi(z)) \cdot X$$

$$= \left[-Y\cdot\frac{1}{\pi} - (1-Y)\frac{1}{1-\pi}\cdot(-1)\right]\cdot\pi\left[1-\pi\right]\cdot X$$

$$= \left[-Y\cdot\frac{1}{\pi} + (1-Y)\frac{1}{1-\pi}\right]\cdot\pi\left[1-\pi\right]\cdot X$$

$$= \left[-Y(1-\pi) + (1-Y)\pi\right]\cdot X$$

$$= \left[-Y+\pi Y+\pi-Y\pi\right]\cdot X$$

$$= \left[-Y+\pi\right]\cdot X$$

$$= \left[-Y+\frac{1}{1+e^{-(\theta_0+\theta_1 X)}}\right]\cdot X = -Y\cdot X + \frac{X}{1+e^{-(\theta_0+\theta_1 X)}}$$

So, $\frac{d^2(NLL)}{d\theta_1^2} = \frac{d}{d\theta_1}\left[-Y\cdot X + \frac{X}{1+e^{-(\theta_0+\theta_1 X)}}\right]$

$$= \frac{d}{d\theta_1}\left[\frac{X}{1+e^{-(\theta_0+\theta_1 X)}}\right]$$

again let $\pi(z) = \frac{1}{1+e^{-z}}$

$$= \frac{d}{d\theta_1}\left[x \cdot \frac{1}{1+e^{-(\theta_0+\theta_1 x)}}\right] = \frac{d}{d\theta_1}\left[x \cdot \pi(z)\right]$$

$$= x \cdot \frac{d\pi}{dz} \cdot \frac{dz}{d\theta_1}$$

$$= x\,\pi(z)\left[1-\pi(z)\right] \cdot x = x^2 \frac{1}{1+e^{-(\theta_0+\theta_1 x)}} \cdot \left[1 - \frac{1}{1+e^{-(\theta_0+\theta_1 x)}}\right]$$

We know that $0 < \frac{1}{1+e^{-(\theta_0+\theta_1 x)}} < 1$ from logistic regression

so $1 - \frac{1}{1+e^{-(\theta_0+\theta_1 x)}} > 0$

so $\frac{d^2(NLL)}{d\theta_1^2} = x^2 \underbrace{\frac{1}{1+e^{-(\theta_0+\theta_1 x)}}}_{>0} \underbrace{\left[1 - \frac{1}{1+e^{-(\theta_0+\theta_1 x)}}\right]}_{>0} > 0$

so $\frac{d^2(NLL)}{d\theta_1^2} > 0$

$$\frac{d(NLL)}{d\theta_0} = \frac{d(NLL)}{d\pi} \cdot \frac{d\pi}{dz} \cdot \frac{dz}{d\theta_0}$$

from solving $\frac{d(NLL)}{d\theta_1}$, we know

$$\frac{d(NLL)}{d\pi} \cdot \frac{d\pi}{dz} = \left[-y + \frac{1}{1+e^{-(\theta_0+\theta_1 x)}}\right]$$

so $\frac{d(NLL)}{d\theta_0} = \left[-y + \frac{1}{1+e^{-(\theta_0+\theta_1 x)}}\right] \cdot \frac{dz}{d\theta_0} = \left[-y + \frac{1}{1+e^{-(\theta_0+\theta_1 x)}}\right] \cdot 1$

$$= -y + \frac{1}{1+e^{-(\theta_0+\theta_1 x)}}$$

$$\text{So, } \frac{d^2(NLL)}{d\theta_0^2} = \frac{d}{d\theta_0}\left[-y + \frac{1}{1+e^{-(\theta_0+\theta_1 x)}}\right]$$

$$= \frac{d}{d\theta_0}\left[\frac{1}{1+e^{-(\theta_0+\theta_1 x)}}\right] = \frac{d}{d\theta_0}[\pi(z)]$$

$$= \frac{d\pi}{dz} \cdot \frac{dz}{d\theta_0} = \pi(z)[1-\pi(z)] \cdot 1 = \pi(z)[1-\pi(z)]$$

$$= \frac{1}{1+e^{-(\theta_0+\theta_1 x)}} \cdot \left(1 - \frac{1}{1+e^{-(\theta_0+\theta_1 x)}}\right)$$

we know from logistic regression

$$0 < \frac{1}{1+e^{-(\theta_0+\theta_1 x)}} < 1, \quad \text{so } 1 - \frac{1}{1+e^{-(\theta_0+\theta_1 x)}} > 0$$

$$\text{So, } \frac{d^2(NLL)}{d\theta_0^2} = \underbrace{\frac{1}{1+e^{-(\theta_0+\theta_1 x)}}}_{>0} \cdot \underbrace{\left[1 - \frac{1}{1+e^{-(\theta_0+\theta_1 x)}}\right]}_{>0} > 0$$

$$\text{So } \frac{d^2(NLL)}{d\theta_0^2} > 0.$$

so for one observation, $\frac{d^2(NLL)}{d\theta_1^2}$ and $\frac{d^2(NLL)}{d\theta_0^2}$ are positive.

so NLL (negative log-likelihood) for one observation is convex.

So the sum of convex functions over multiple individual observations is convex.

2) For logistic regression with individual observations:

$$L_1 = \text{likelihood} = \prod_{i=1}^{N_1} \pi_i^{Y_i} (1-\pi_i)^{1-Y_i}$$

π - probability Y - outcome

For logistic regression with grouped observations:

$$L_2 = \prod_{j=1}^{N_2} \prod_{i=1}^{N_1} \pi_i^{Y_i} (1-\pi_i)^{1-Y_i}$$

Y	probability π
0	
1	
⋮	
N_2	

So, we can think of Y as a binomial random variable $Y \sim \text{Binomial}(N_2, \pi)$ as a sum of N_2 Bernoulli random variables $Y' \sim \text{Bernoulli}(\pi)$.

So probability mass function $f(y) = \binom{N_2}{y} \pi^y (1-\pi)^{N_2-y}$

So $L_2 = \binom{N_2}{y} \pi^y (1-\pi)^{N_2-y}$

So, to find the maximum likelihood estimates, we take log likelihood of L_1 and L_2

$$\log L_1 = \log \prod_{i=1}^{N_1} \pi_i^{Y_i} (1-\pi_i)^{1-Y_i}$$

$$= \sum_{i=1}^{N_1} \left[Y_i \log \pi_i + (1-Y_i) \log (1-\pi_i) \right]$$

$$\log L_2 = \log \prod_{i=1}^{N_1} \left[\binom{N_2}{Y} \pi_i^{Y_i} (1-\pi_i)^{(N_2-Y_i)} \right]$$

$$= \sum_{i=1}^{N_1} \binom{N_2}{Y} + \sum_{i=1}^{N_1} \left[Y_i \pi_i + (N_2 - Y_i)(1-\pi_i) \right].$$

does not depend on π_i, can ignore

$Y_i = \sum_{j=1}^{N_2} Y_{ij}$, where $Y_{ij} = \begin{cases} 1 & \pi \\ 0 & 1-\pi \end{cases}$

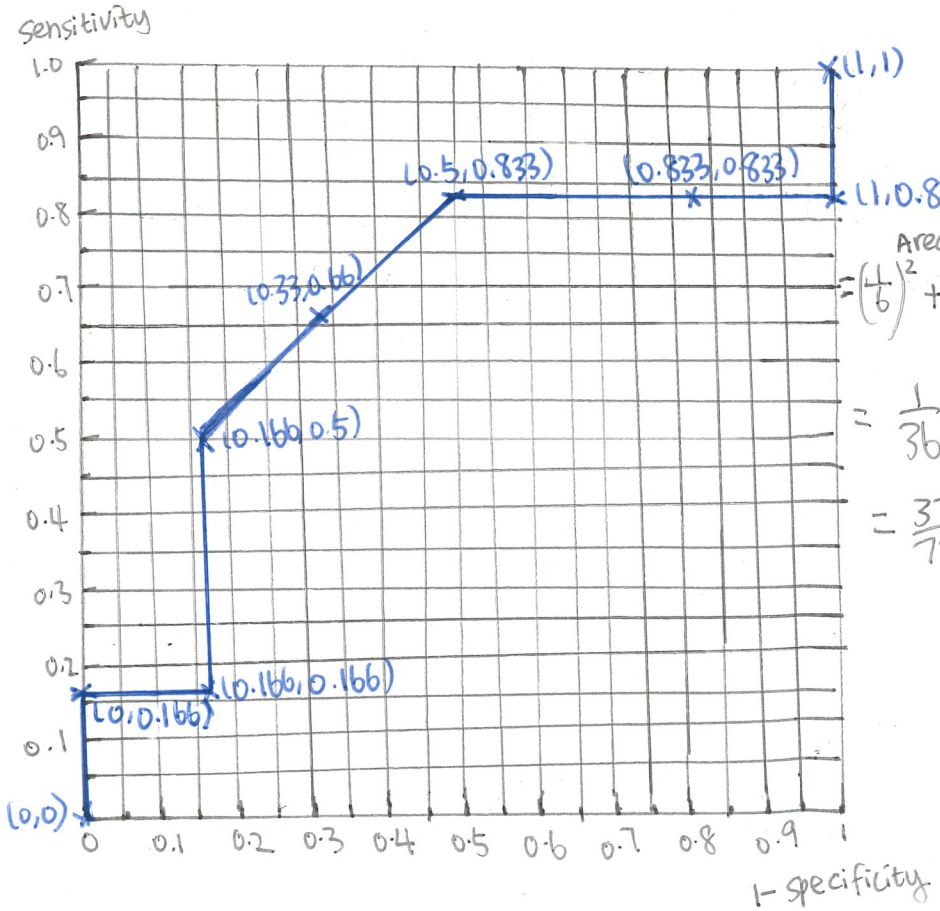
because $Y \sim \text{Binomial}(N_2, \pi)$ is a sum of N_2 ~~Bernoulli random variables~~ $Y' \sim \text{Bernoulli}(\pi)$

so, $\log L_2 = \text{const} + \sum_{j=1}^{N_2} \sum_{i=1}^{N_1} \left[Y_i \log(\pi_i) + (1-Y_i) \log(1-\pi_i) \right]$

$$= \text{const} + \sum_{j=1}^{N_2} \log L_1$$

So, individual and grouped ~~observations~~ will give the same maximum likelihood estimates of the parameters if we know N_2.

cutoff	TP	FP	FN	TN	N (=FP+TN)	P (=TP+FN)	1−specificity (= 1 − TN/N)	sensitivity (= TP/P)
0.05	5	6	1	0	6	6	1	0.8333 (5/6)
0.10	5	5	1	1	6	6	0.8333 (5/6)	0.8333 (5/6)
0.26	5	3	1	3	6	6	0.5	0.8333
0.32	4	2	2	4	6	6	0.3333 (1/3)	0.6666 (2/3)
0.60	3	1	3	5	6	6	0.1666 (1/6)	0.5
0.80	1	1	5	5	6	6	0.1666 (1/6)	0.1666 (1/6)
0.92	1	0	5	6	6	6	0	0.1666
0.96	0	0	6	6	6	6	0	0
0.005	6	6	0	0	6	6	1	1



Q3:

Area under ROC

$= \left(\frac{1}{6}\right)^2 + 0.5 \times \frac{5}{6} + \frac{0.5 \times \frac{5}{6}}{2} \times \frac{1}{3}$

$= \frac{1}{36} + \frac{5}{12} + \frac{5}{72}$

$= \frac{37}{72} = 0.5138$

5) a)

X_1	X_2	X_3	Y
0	1	1	1
1	0	0	1
1	1	0	1
0	1	1	1
1	1	0	0
1	0	1	0
0	0	1	0

With naive Bayes assumption:

$$P(Y=1 \mid X_1=1, X_2=0, X_3=1)$$

$$= \frac{P(X_1=1, X_2=0, X_3=1 \mid Y=1) P(Y=1)}{P(X_1=1, X_2=0, X_3=1 \mid Y=0) P(Y=0) + P(X_1=1, X_2=0, X_3=1 \mid Y=1) P(Y=1)}$$

$$= \frac{P(X_1=1 \mid Y=1) P(X_2=0 \mid Y=1) P(X_3=1 \mid Y=1) P(Y=1)}{P(X_1=1 \mid Y=0) P(X_2=0 \mid Y=0) P(X_3=1 \mid Y=0) P(Y=0) + P(X_1=1 \mid Y=1) P(X_2=0 \mid Y=1) P(X_3=1 \mid Y=1) P(Y=1)}$$

$$= \frac{\frac{1}{2} \cdot \frac{1}{4} \cdot \frac{1}{2} \cdot \frac{4}{7}}{\frac{2}{3} \cdot \frac{2}{3} \cdot \frac{2}{3} \cdot \frac{3}{7} + \frac{1}{2} \cdot \frac{1}{4} \cdot \frac{1}{2} \cdot \frac{4}{7}}$$

$$= \frac{\frac{1}{28}}{\frac{8}{63} + \frac{1}{28}} = \frac{9}{41} = 0.2195$$

$$P(Y=1 \mid X_1=1, X_2=1, X_3=1)$$

$$= \frac{P(X_1=1, X_2=1, X_3=1 \mid Y=1) P(Y=1)}{P(X_1=1, X_2=1, X_3=1 \mid Y=0) P(Y=0) + P(X_1=1, X_2=1, X_3=1 \mid Y=1) P(Y=1)}$$

$$= \frac{P(X_1=1 \mid Y=1) P(X_2=1 \mid Y=1) P(X_3=1 \mid Y=1) P(Y=1)}{P(X_1=1 \mid Y=0) P(X_2=1 \mid Y=0) P(X_3=1 \mid Y=0) P(Y=0) + P(X_1=1 \mid Y=1) P(X_2=1 \mid Y=1) P(X_3=1 \mid Y=1) P(Y=1)}$$

$$= \frac{\frac{1}{2} \cdot \frac{3}{4} \cdot \frac{1}{2} \cdot \frac{4}{7}}{\frac{2}{3} \cdot \frac{1}{3} \cdot \frac{2}{3} \cdot \frac{3}{7} + \frac{1}{2} \cdot \frac{3}{4} \cdot \frac{1}{2} \cdot \frac{4}{7}}$$

$$= \frac{\frac{3}{28}}{\frac{4}{63} + \frac{3}{28}} = \frac{27}{43}$$

$$= 0.6279$$

5 a)

Number of parameters:

these add to 1 so considered as one parameter: $P(X_1=0|Y=0)$, $P(X_1=1|Y=0)$

$P(X_1=0|Y=1)$, $P(X_1=1|Y=1)$ } one parameter

add to 1: $P(X_2=0|Y=0)$, $P(X_2=1|Y=0)$

Considered as one parameter

$P(X_2=0|Y=1)$, $P(X_2=1|Y=1)$ } one parameter

one parameter: $P(X_3=0|Y=0)$, $P(X_3=1|Y=0)$

$P(X_3=0|Y=1)$, $P(X_3=1|Y=1)$ } one parameter

so, number of parameters

$= 2n+1 = 2\times3+1 = 6+1 = 7$

$n=3$ for X_1, X_2, X_3

number of parameter for $P(Y)$ is 1

b) without Naïve Bayes:

$$P(Y=1 \mid X_1=1, X_2=0, X_3=1)$$

$$= \frac{P(X_1=1, X_2=0, X_3=1 \mid Y=1)\, P(Y=1)}{P(X_1=1, X_2=0, X_3=1 \mid Y=0)\, P(Y=0) + P(X_1=1, X_2=0, X_3=1 \mid Y=1) \cdot P(Y=1)}$$

$$= \frac{0 \cdot \frac{4}{7}}{\frac{1}{3} \cdot \frac{3}{7} + 0} = 0$$

$$P(Y=1 \mid X_1=1, X_2=1, X_3=1)$$

$$= \frac{P(X_1=1, X_2=1, X_3=1 \mid Y=1)\, P(Y=1)}{P(X_1=1, X_2=1, X_3=1 \mid Y=0)\, P(Y=0) + P(X_1=1, X_2=1, X_3=1 \mid Y=1) \cdot P(Y=1)}$$

$$= \frac{0 \cdot \frac{4}{7}}{0+0}$$ undefined.

5b) Without Naive Bayes

$P(X_1=0, X_2=0, X_3=0 \mid Y=0)$

$P(X_1=0, X_2=0, X_3=1 \mid Y=0)$

We have $2^3 = 8$ combinations.

0	1	0
1	0	0
0	1	1
1	0	1
1	1	0
1	1	1

So, number of parameters

$$= (2^3 - 1) \times 2 + 1 = 7 \times 2 + 1 = 15.$$

this constraints the number of parameters because frequencies are probabilities. they have to add up to 1

number of parameters of $P(Y)$ is 1

we have $P(X_1, X_2, X_3 \mid Y=0)$ and $P(X_1, X_2, X_3 \mid Y=1)$. so we times it by 2

Question 4a

In [ ]:

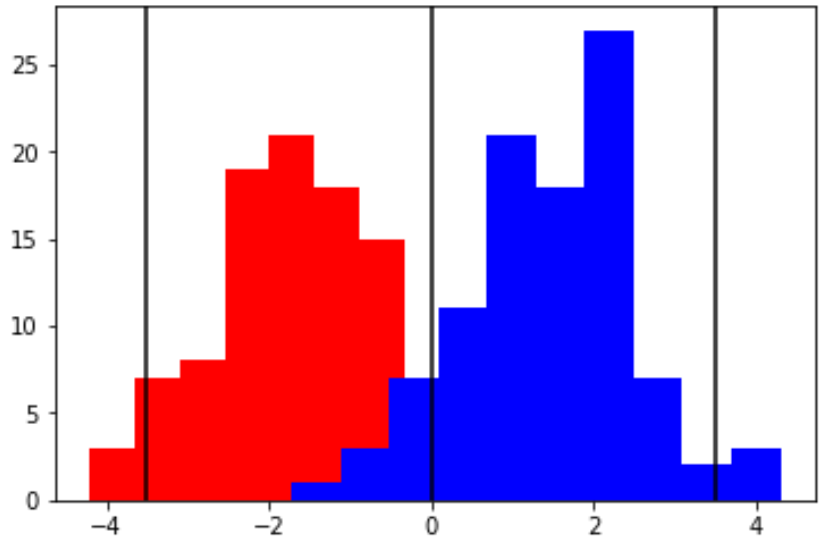
```
import numpy as np
import matplotlib.pyplot as plt
from numpy.linalg import inv
import math
```

In [3]:

```
X_class0 = np.random.normal(-1.5,1,100)
X_class1 = np.random.normal(1.5,1,100)
```

In [4]:

```
plt.hist(X_class0, color='red')
plt.hist(X_class1, color='blue')
plt.axvline(x=0, color='black')
plt.axvline(x=-3.5, color='black')
plt.axvline(x=3.5, color='black')
plt.show()
```



Question 4a

In [ ]:

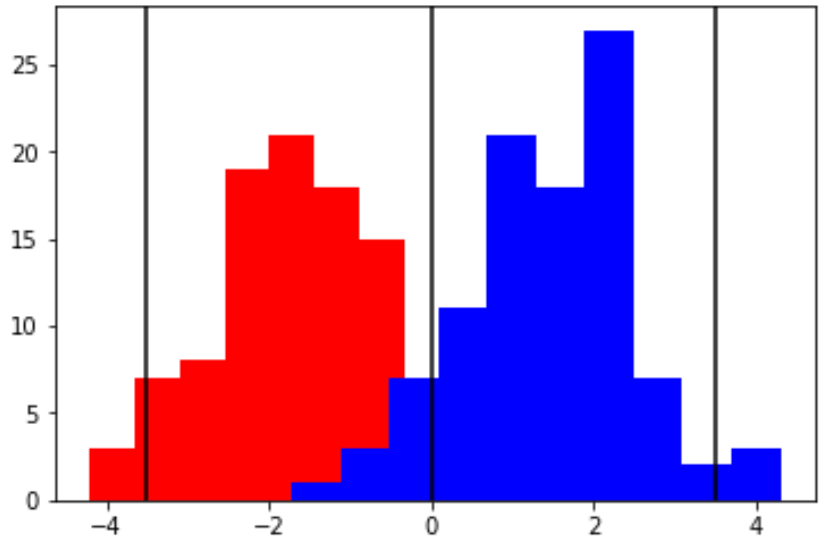
```
import numpy as np
import matplotlib.pyplot as plt
from numpy.linalg import inv
import math
```

In [3]:

```
X_class0 = np.random.normal(-1.5,1,100)
X_class1 = np.random.normal(1.5,1,100)
```

In [4]:

```
plt.hist(X_class0, color='red')
plt.hist(X_class1, color='blue')
plt.axvline(x=0, color='black')
plt.axvline(x=-3.5, color='black')
plt.axvline(x=3.5, color='black')
plt.show()
```



In [5]:

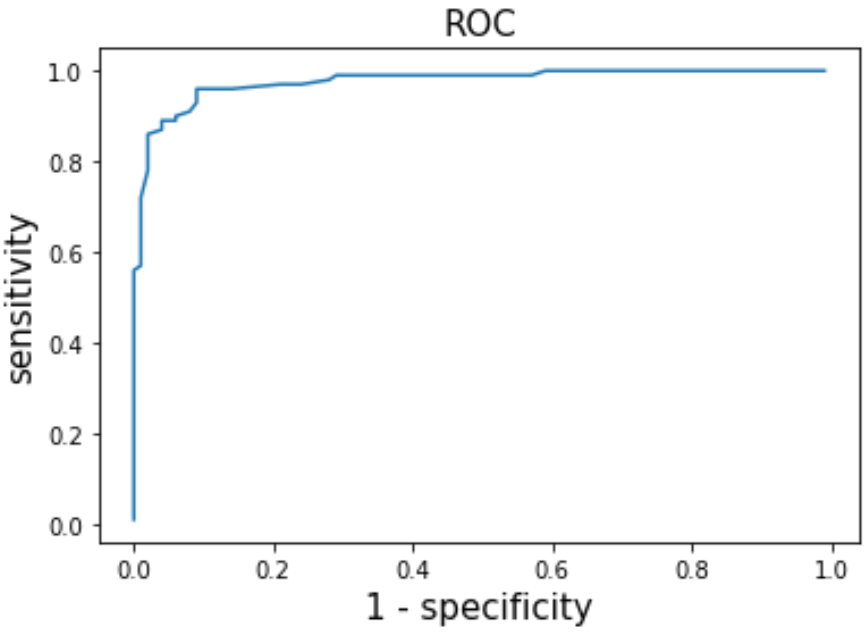
```
one_minus_specificity=[]
sensitivity=[]
precision=[]
cutoff=np.linspace(min(X_class0),max(X_class1)-0.000001,100)

for j in range(len(cutoff)):
    TP=0
    FP=0
    FN=0
    TN=0

    TP += len(X_class1[X_class1 > cutoff[j]])
    FP += len(X_class0[X_class0 > cutoff[j]])
    FN += len(X_class1[X_class1 <= cutoff[j]])
    TN += len(X_class0[X_class0 <= cutoff[j]])
    N=FP+TN
    P=TP+FN


    one_minus_specificity.append(1-(TN/N))
    sensitivity.append(TP/P)
    precision.append(TP/(TP+FP))
```

In [6]:

```
plt.ylabel('sensitivity', fontsize = 15)
plt.xlabel('1 - specificity', fontsize=15)
plt.title("ROC", fontsize=15)
plt.plot(one_minus_specificity,sensitivity)
```

Out[6]:

```
[<matplotlib.lines.Line2D at 0x120939dd8>]
```



In [5]:

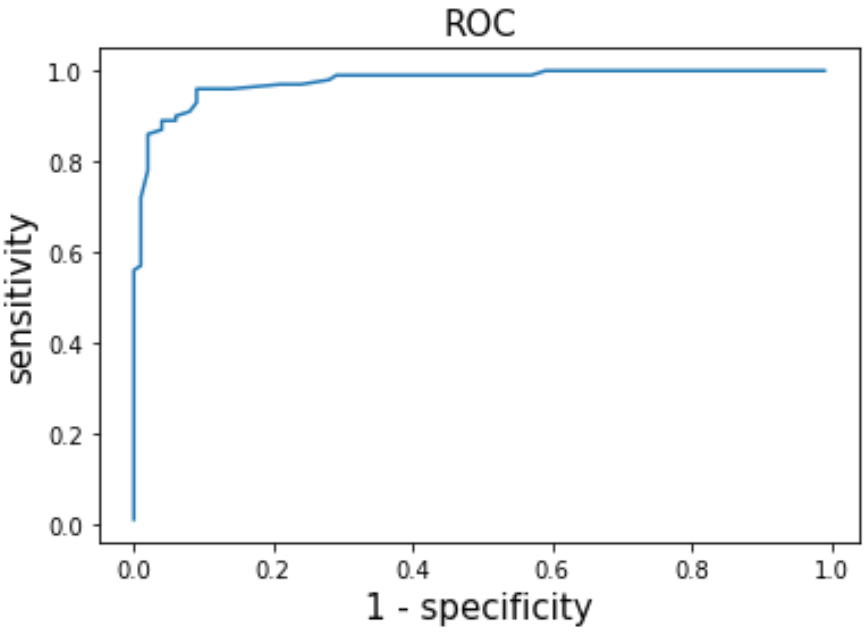
```
one_minus_specificity=[]
sensitivity=[]
precision=[]
cutoff=np.linspace(min(X_class0),max(X_class1)-0.000001,100)

for j in range(len(cutoff)):
    TP=0
    FP=0
    FN=0
    TN=0

    TP += len(X_class1[X_class1 > cutoff[j]])
    FP += len(X_class0[X_class0 > cutoff[j]])
    FN += len(X_class1[X_class1 <= cutoff[j]])
    TN += len(X_class0[X_class0 <= cutoff[j]])
    N=FP+TN
    P=TP+FN


    one_minus_specificity.append(1-(TN/N))
    sensitivity.append(TP/P)
    precision.append(TP/(TP+FP))
```

In [6]:

```
plt.ylabel('sensitivity', fontsize = 15)
plt.xlabel('1 - specificity', fontsize=15)
plt.title("ROC", fontsize=15)
plt.plot(one_minus_specificity,sensitivity)
```

Out[6]:

```
[<matplotlib.lines.Line2D at 0x120939dd8>]
```

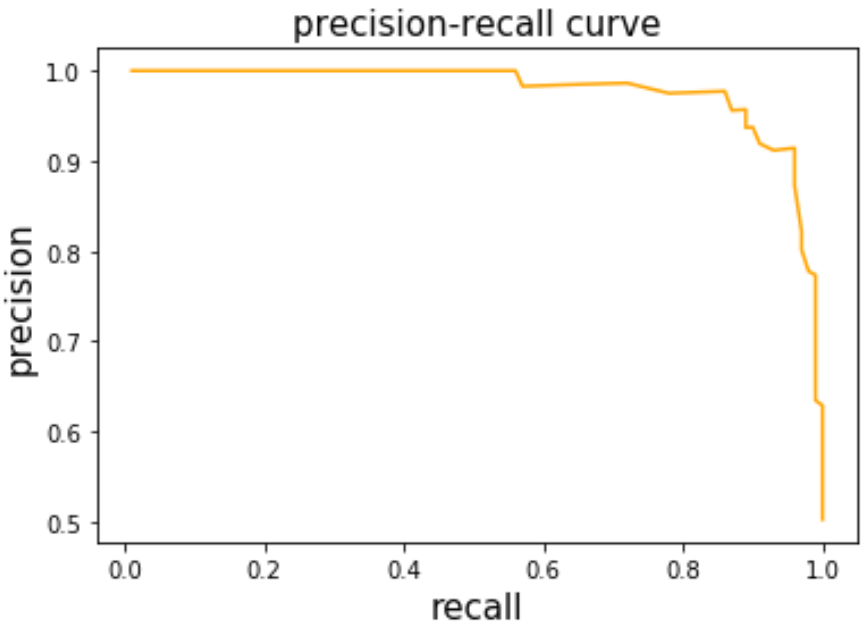


In [7]:

```
plt.ylabel('precision', fontsize = 15)
plt.xlabel('recall', fontsize=15)
plt.title("precision-recall curve", fontsize=15)
plt.plot(sensitivity,precision, color='orange')
```

Out[7]:

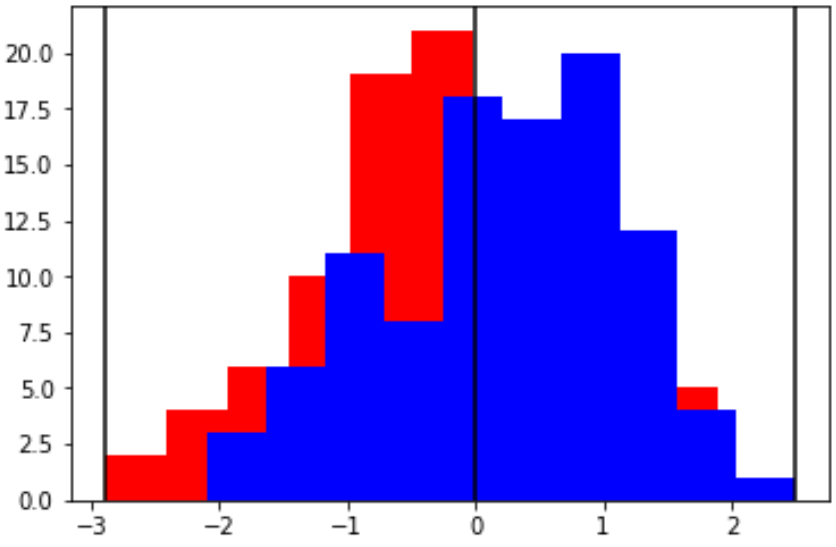
```
[<matplotlib.lines.Line2D at 0x1209dd588>]
```

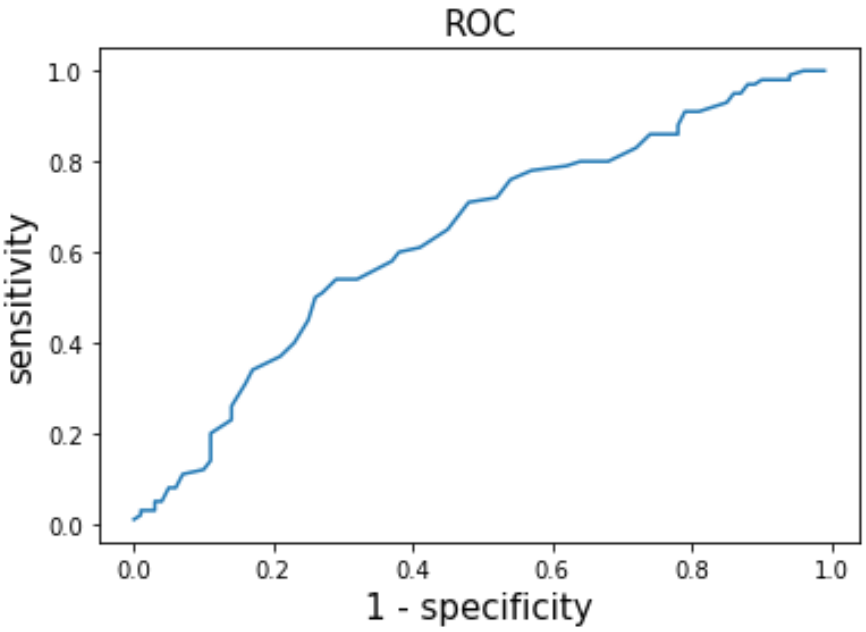


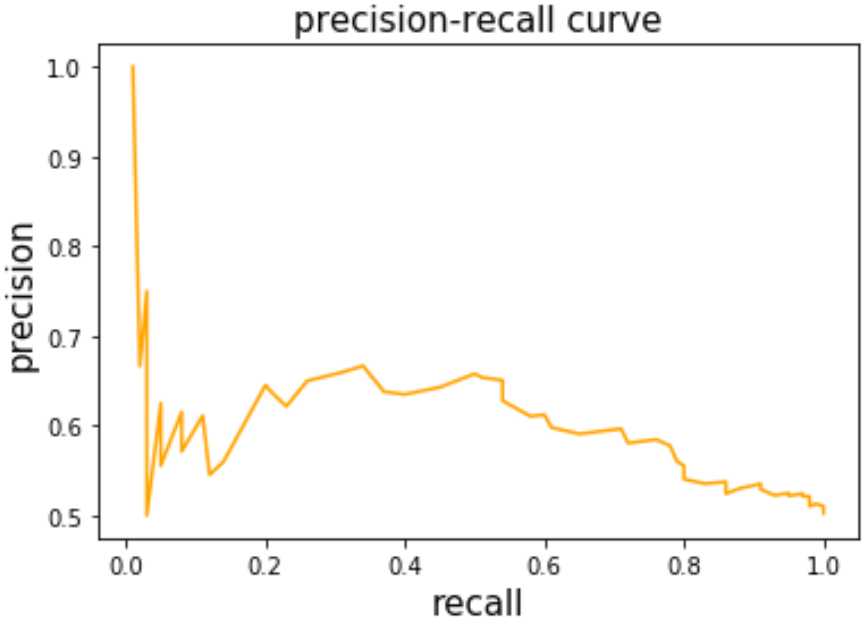
Question 4b

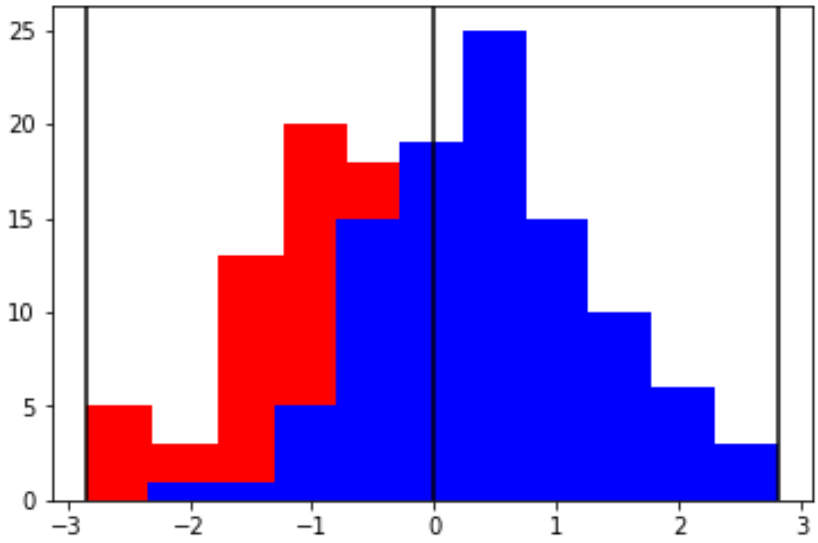
In [15]:

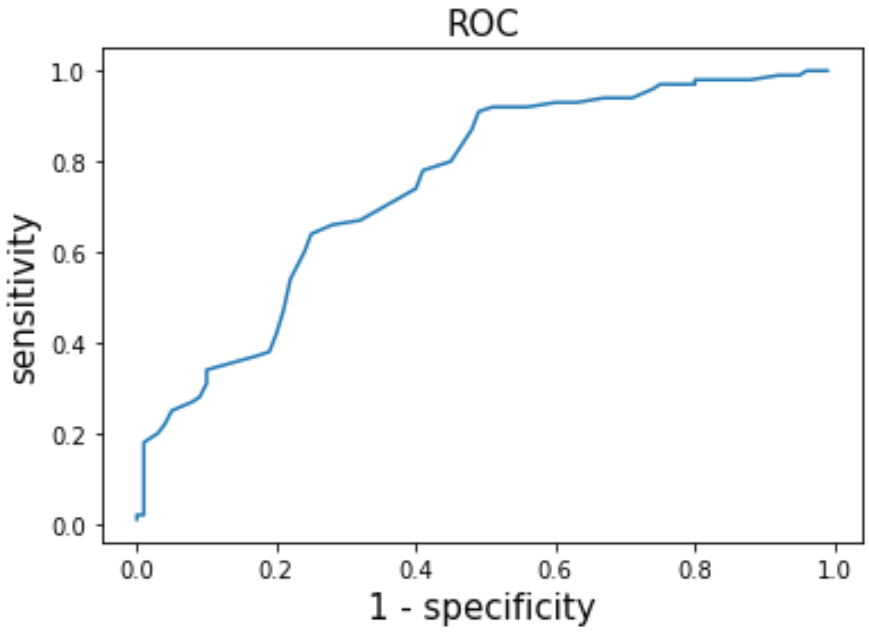
```
mean=np.linspace(0.1,3,10)

for i in range(10):
    X_class0 = np.random.normal(-mean[i],1,100)
    X_class1 = np.random.normal(mean[i],1,100)

    one_minus_specificity=[]
    sensitivity=[]
    precision=[]
    cutoff=np.linspace(min(min(X_class0),min(X_class1)),max(max(X_class1),max(X_
class0))-0.000000001,70)

    plt.hist(X_class0, color='red')
    plt.hist(X_class1, color='blue')
    plt.axvline(x=0, color='black')
    plt.axvline(x=min(min(X_class0),min(X_class1)), color='black')
    plt.axvline(x=max(max(X_class1),max(X_class0))-0.000000001, color='black')
    plt.show()


    for j in range(len(cutoff)):
        TP=0
        FP=0
        FN=0
        TN=0
        TP += len(X_class1[X_class1 > cutoff[j]])
        FP += len(X_class0[X_class0 > cutoff[j]])
        FN += len(X_class1[X_class1 <= cutoff[j]])
        TN += len(X_class0[X_class0 <= cutoff[j]])
        N=FP+TN
        P=TP+FN

        one_minus_specificity.append(1-(TN/N))
        sensitivity.append(TP/P)
        precision.append(TP/(TP+FP))


    plt.ylabel('sensitivity', fontsize = 15)
    plt.xlabel('1 - specificity', fontsize=15)
    plt.title("ROC", fontsize=15)
    plt.plot(one_minus_specificity,sensitivity)
    plt.show()

    plt.ylabel('precision', fontsize = 15)
    plt.xlabel('recall', fontsize=15)
    plt.title("precision-recall curve", fontsize=15)
    plt.plot(sensitivity,precision, color='orange')
    plt.show()
```

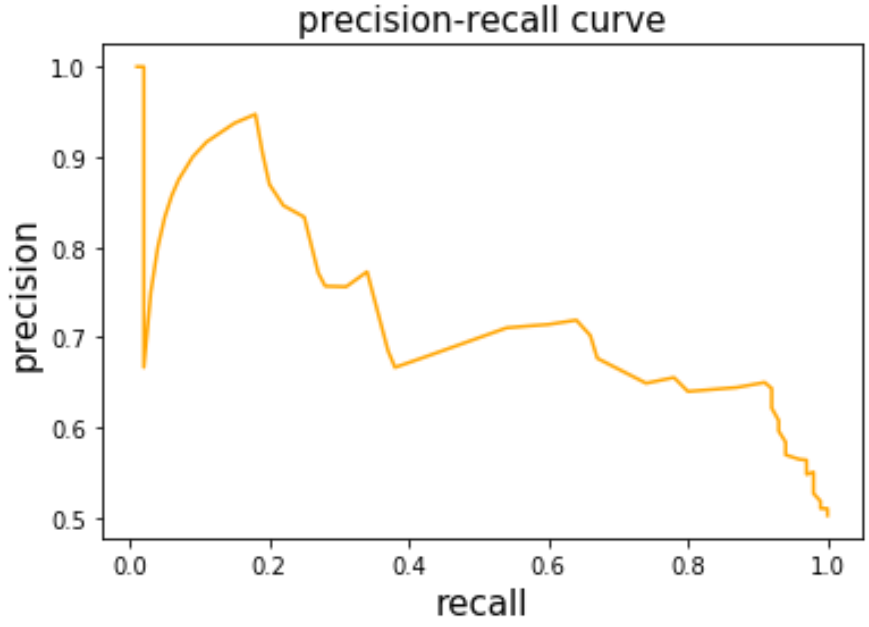

20.0
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
−3
−2
−1
0
1
2

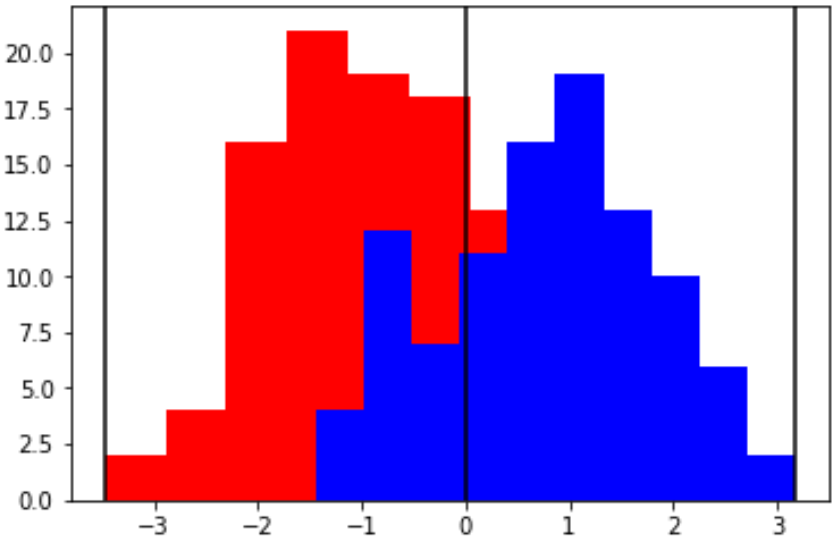

ROC
sensitivity
1 - specificity
0.0
0.2
0.4
0.6
0.8
1.0

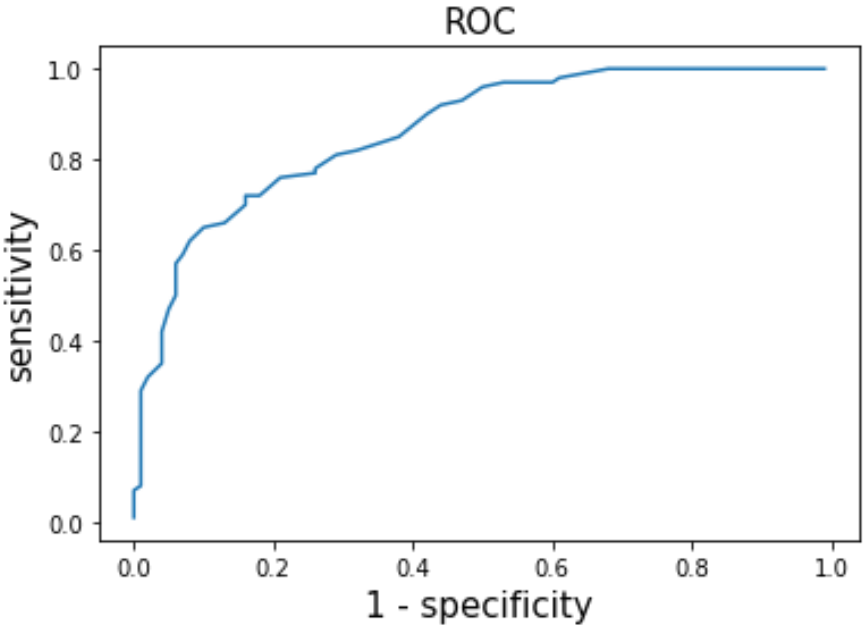

precision-recall curve
precision
recall
1.0
0.9
0.8
0.7
0.6
0.5
0.0
0.2
0.4
0.6
0.8
1.0

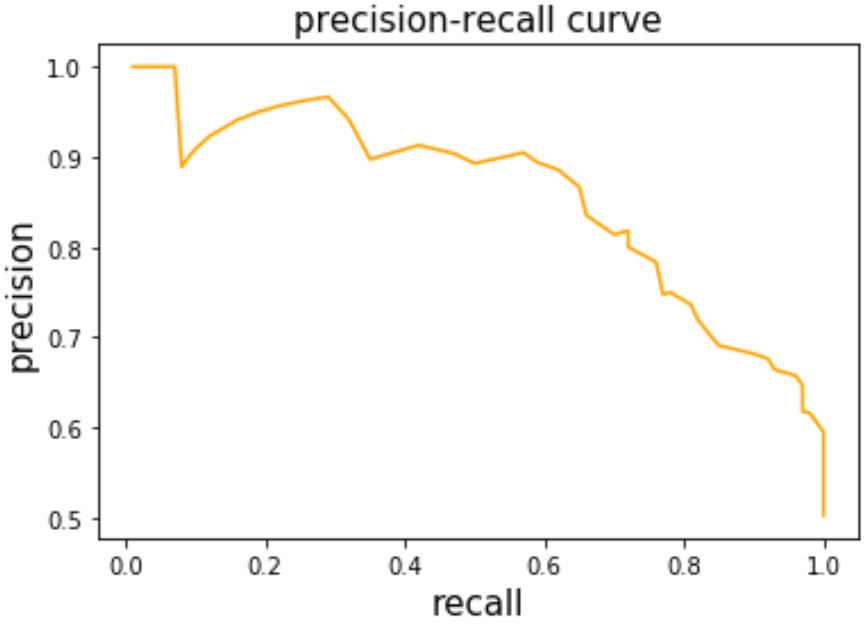

25
20
15
10
5
0
−3
−2
−1
0
1
2
3

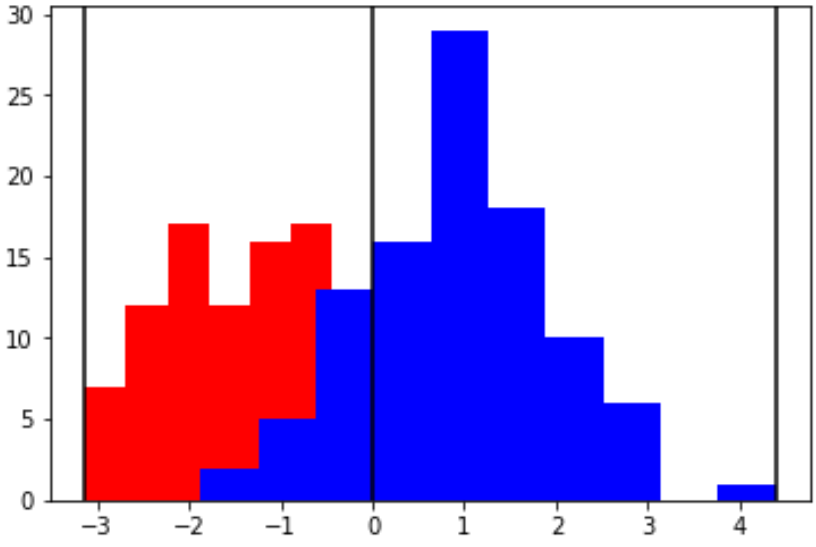

ROC
sensitivity
1 - specificity
0.0
0.2
0.4
0.6
0.8
1.0

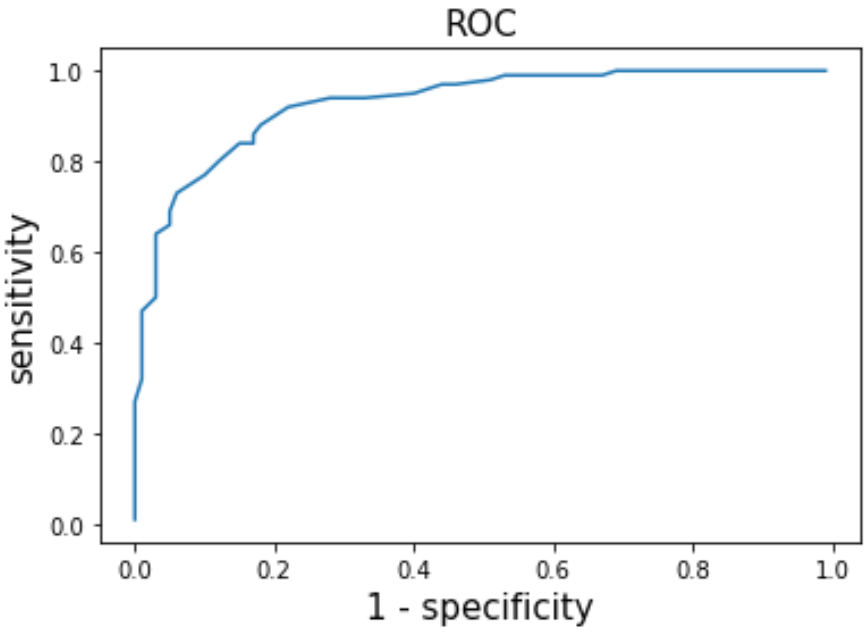

precision-recall curve
precision
recall
1.0
0.9
0.8
0.7
0.6
0.5
0.0
0.2
0.4
0.6
0.8
1.0

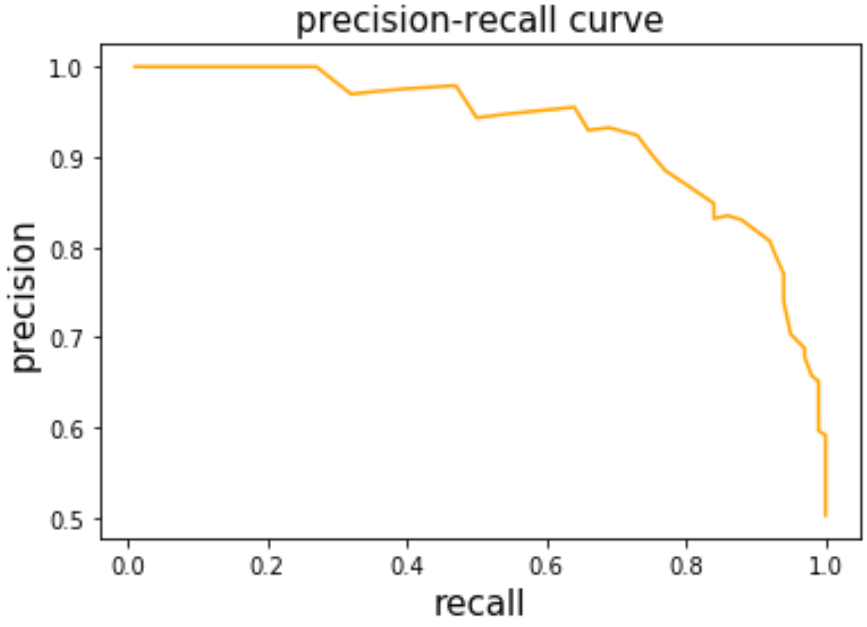

20.0
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
−3
−2
−1
0
1
2
3

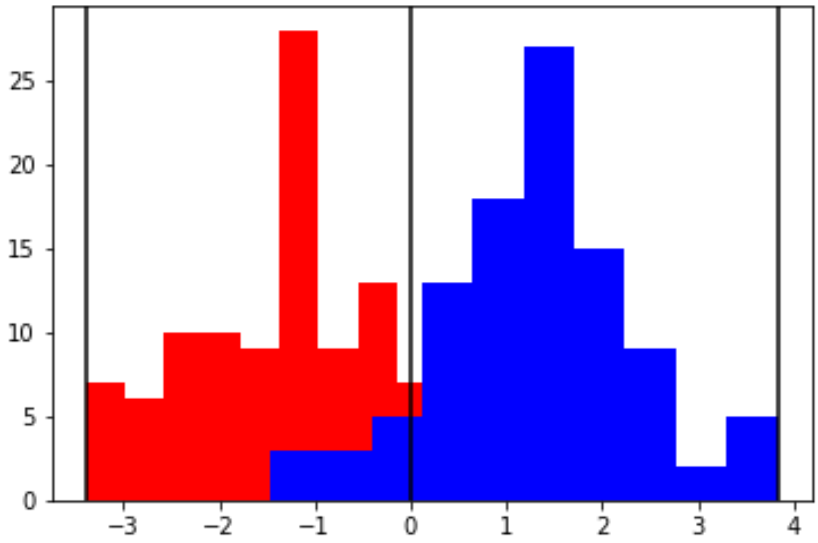

ROC
sensitivity
1 - specificity
0.0
0.2
0.4
0.6
0.8
1.0

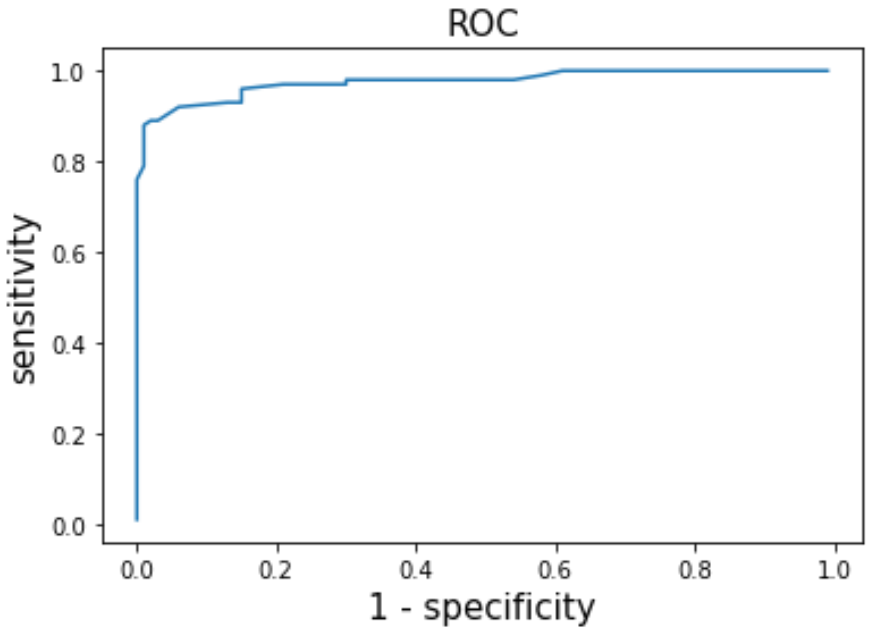

precision-recall curve
precision
recall
1.0
0.9
0.8
0.7
0.6
0.5
0.0
0.2
0.4
0.6
0.8
1.0

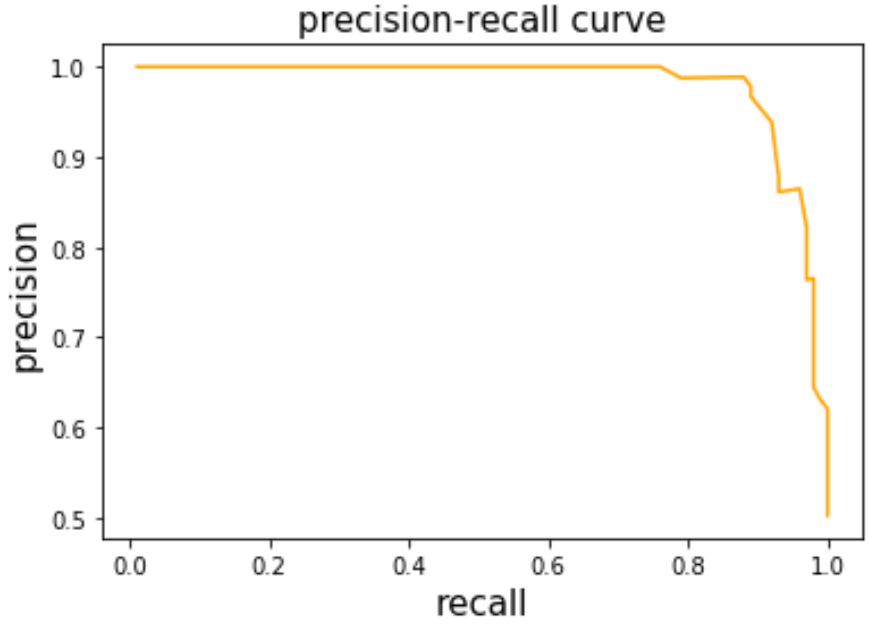

30
25
20
15
10
5
0
−3
−2
−1
0
1
2
3
4

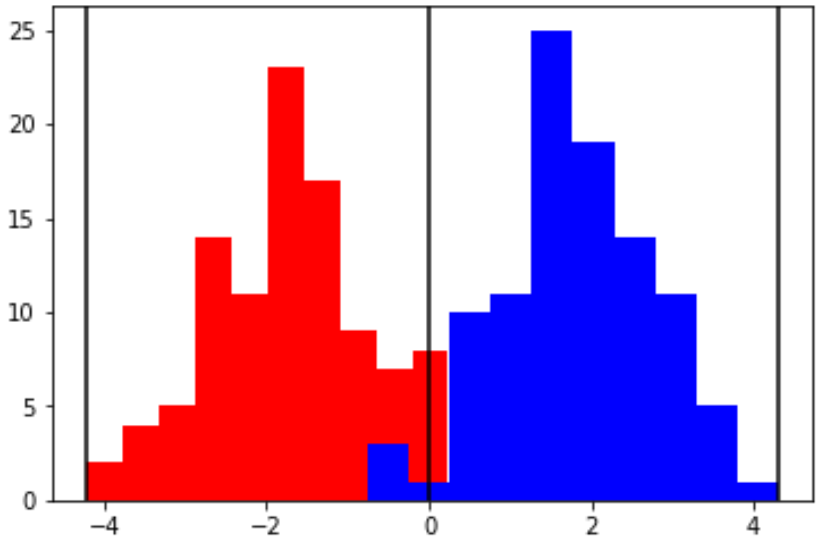

ROC
sensitivity
1 - specificity
0.0
0.2
0.4
0.6
0.8
1.0

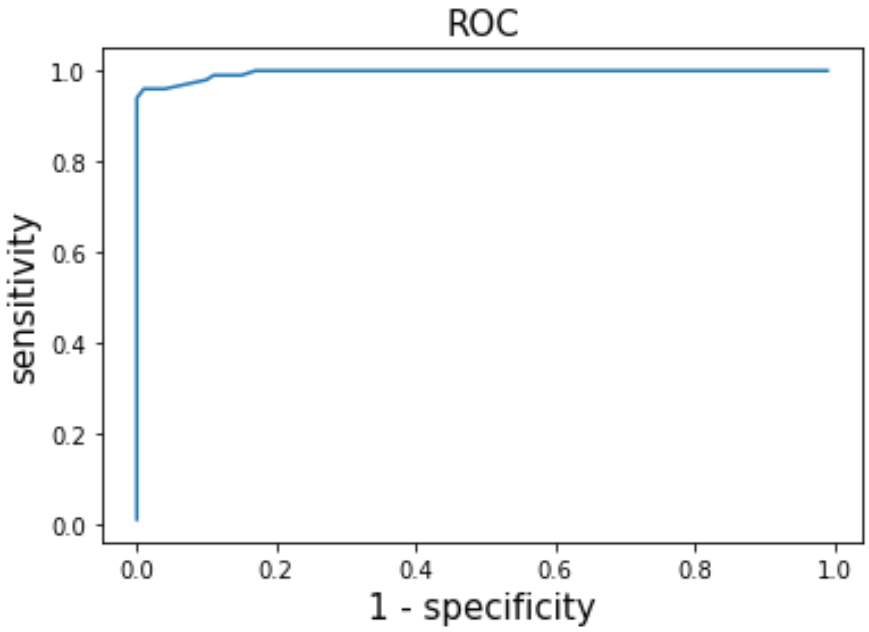

precision-recall curve
precision
recall
1.0
0.9
0.8
0.7
0.6
0.5
0.0
0.2
0.4
0.6
0.8
1.0

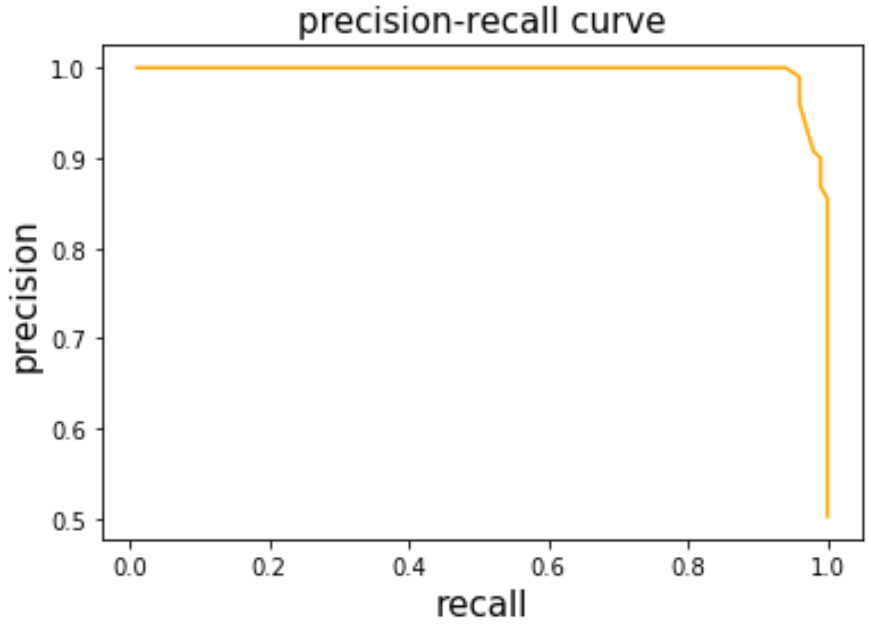

25
20
15
10
5
0
−3
−2
−1
0
1
2
3
4

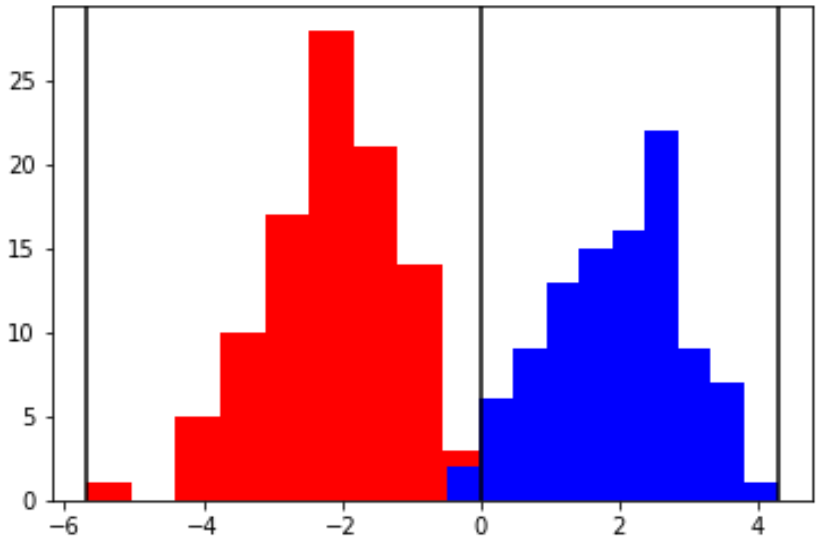

ROC
sensitivity
1 - specificity
0.0
0.2
0.4
0.6
0.8
1.0

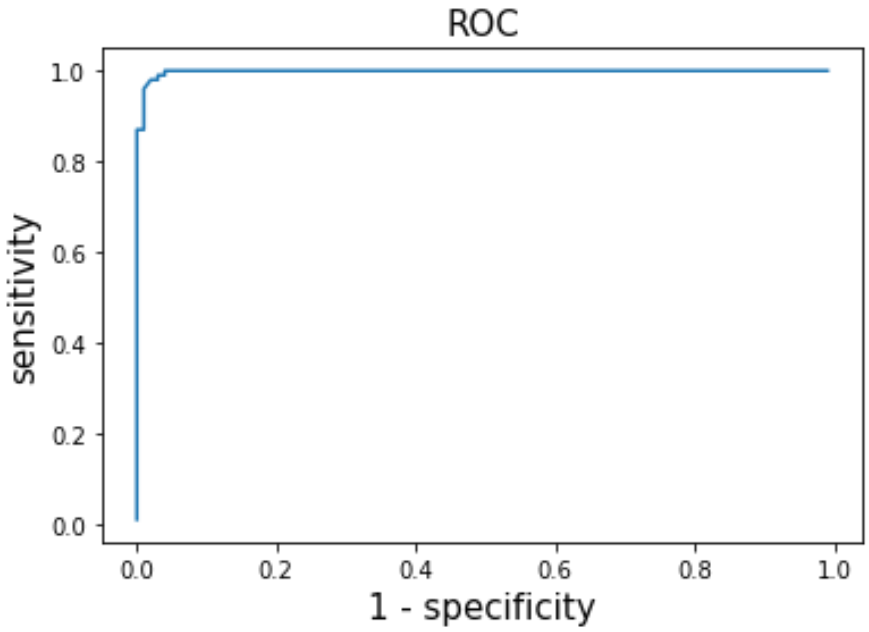

precision-recall curve
precision
recall
1.0
0.9
0.8
0.7
0.6
0.5
0.0
0.2
0.4
0.6
0.8
1.0

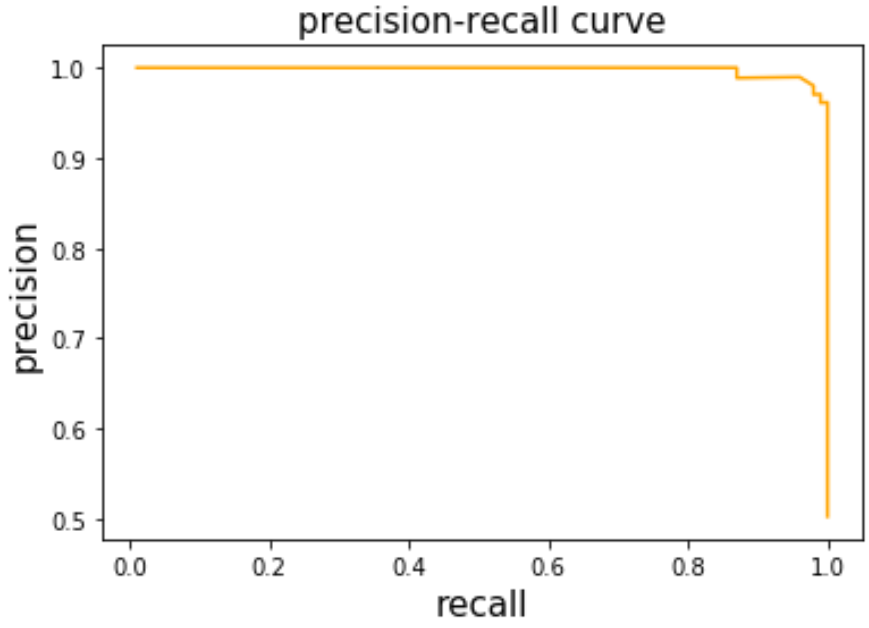

25
20
15
10
5
0
−4
−2
0
2
4

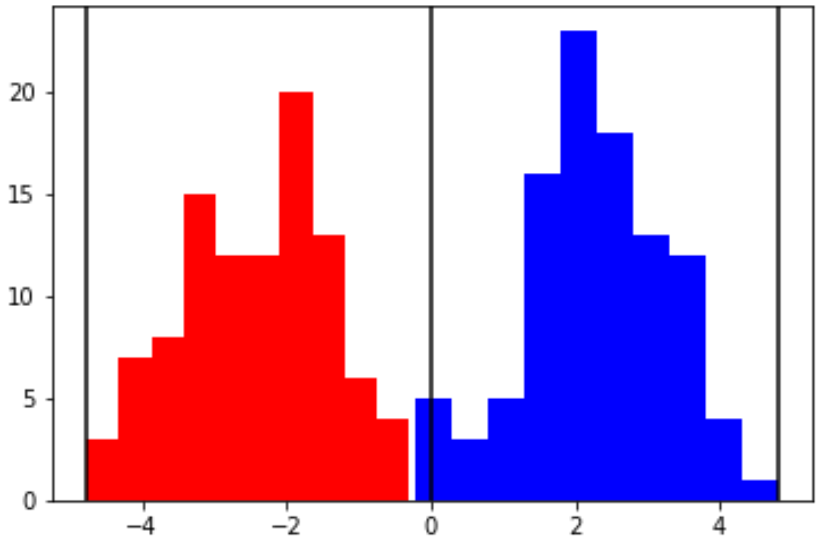

ROC
sensitivity
1 - specificity
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

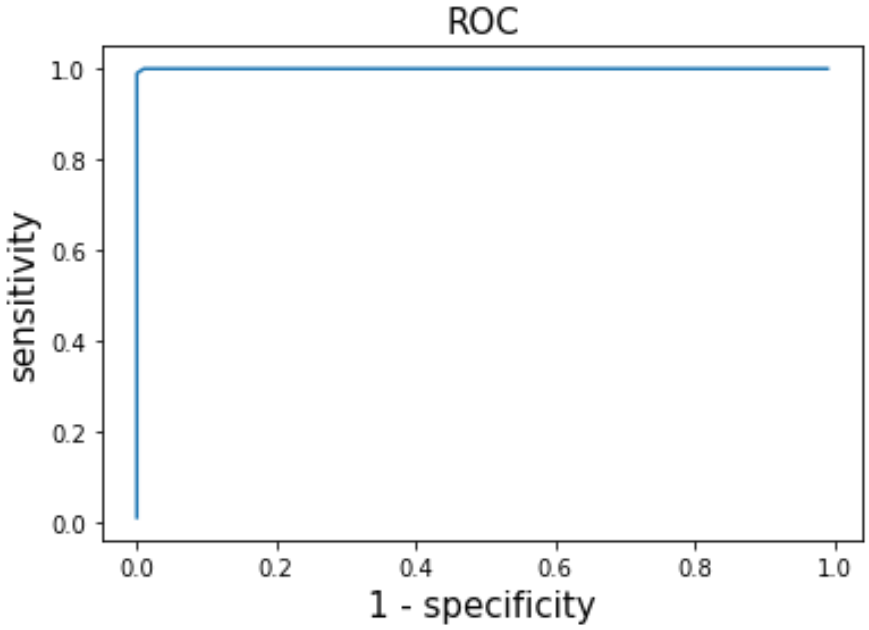

precision-recall curve
precision
recall
1.0
0.9
0.8
0.7
0.6
0.5
0.0
0.2
0.4
0.6
0.8
1.0

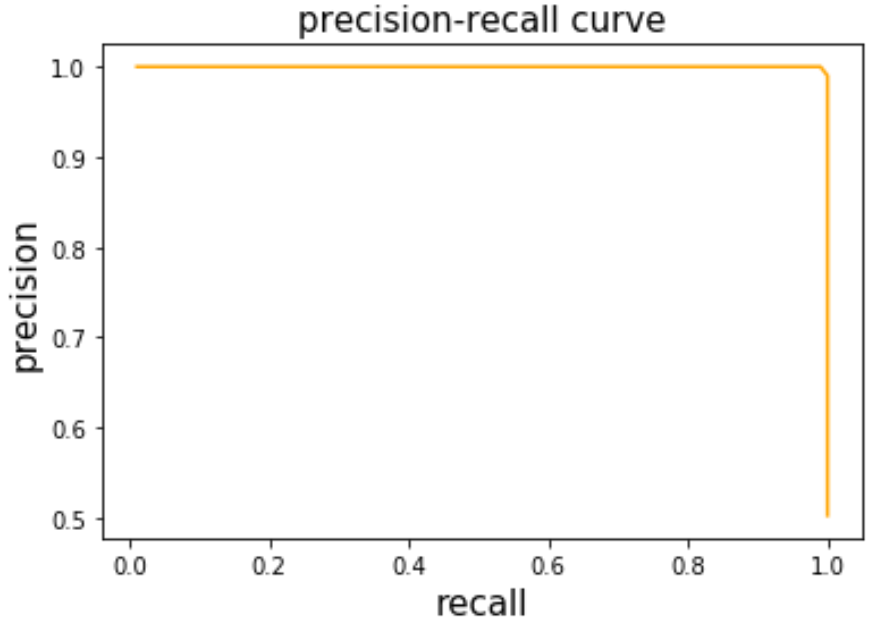

25
20
15
10
5
0
−6
−4
−2
0
2
4

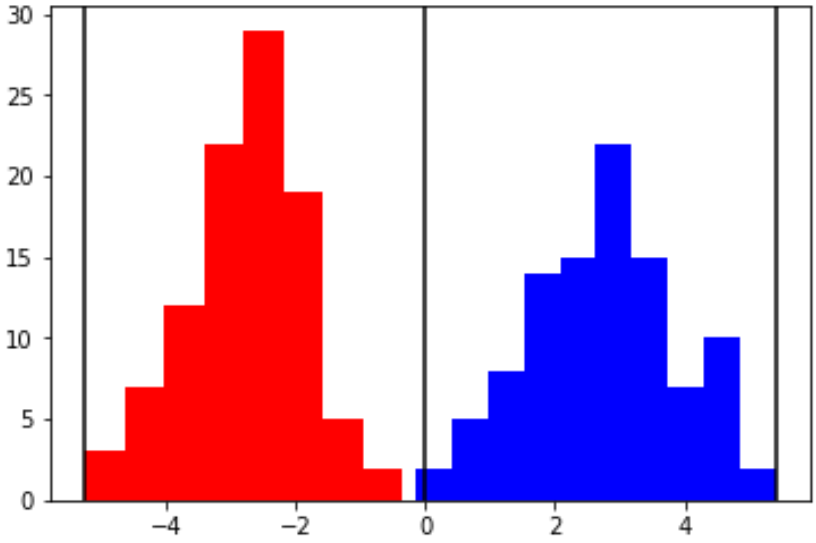

ROC
sensitivity
1 - specificity
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

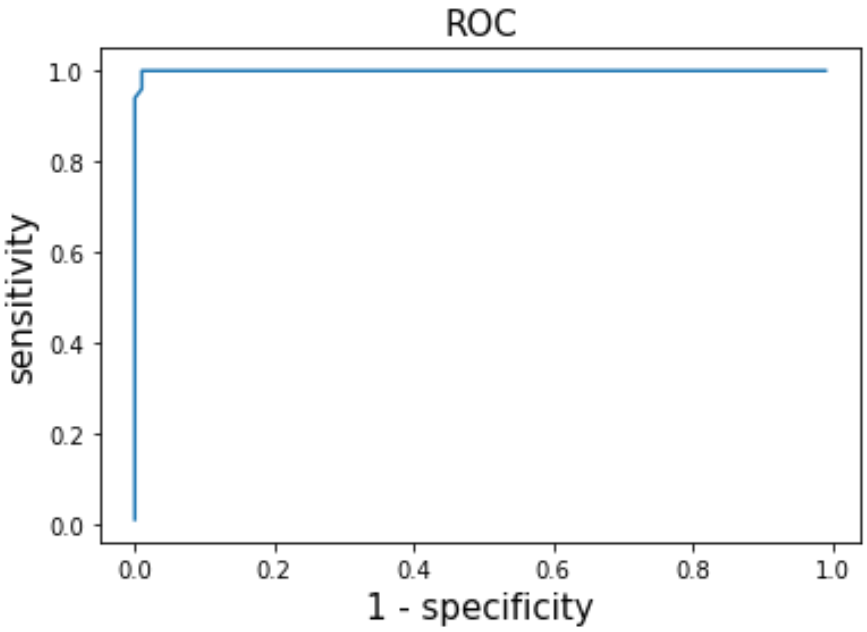

precision-recall curve
precision
recall
1.0
0.9
0.8
0.7
0.6
0.5
0.0
0.2
0.4
0.6
0.8
1.0

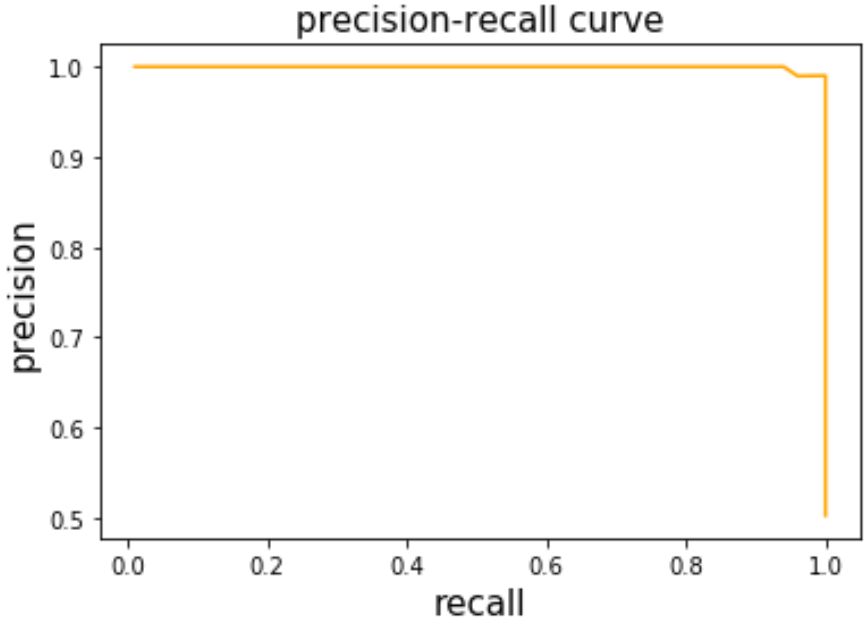

20
15
10
5
0
−4
−2
0
2
4

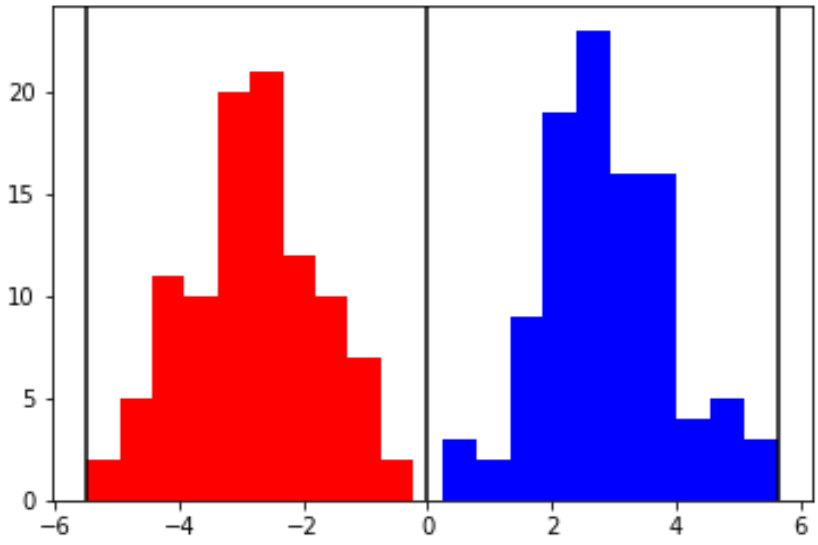

ROC
sensitivity
1 - specificity
0.0
0.2
0.4
0.6
0.8
1.0

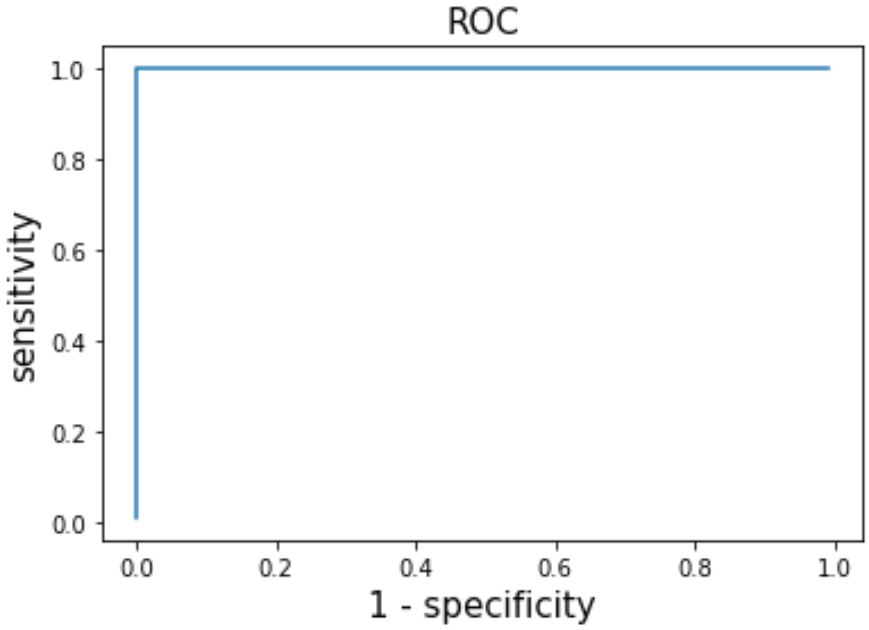

precision-recall curve
precision
recall
1.0
0.9
0.8
0.7
0.6
0.5
0.0
0.2
0.4
0.6
0.8
1.0

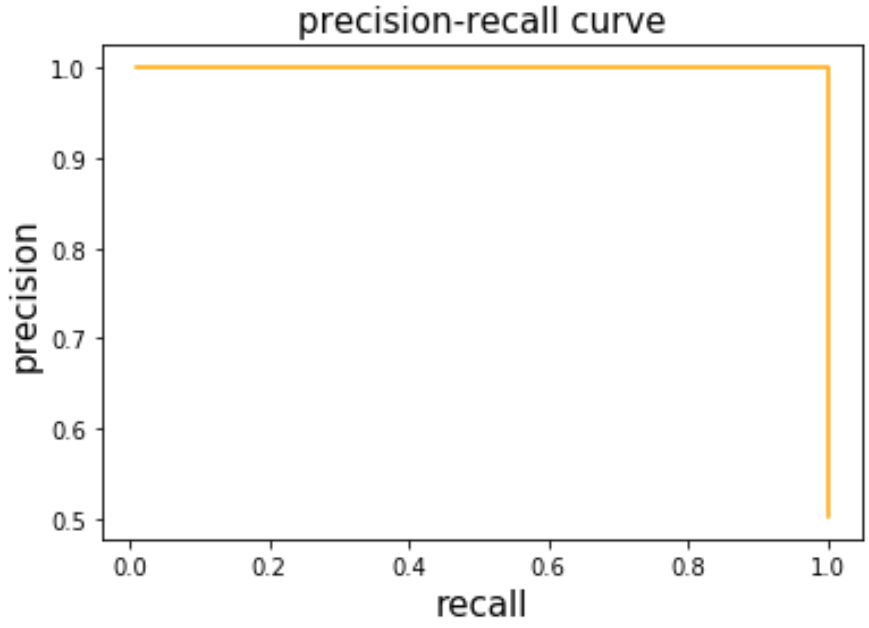

30
25
20
15
10
5
0
−4
−2
0
2
4


ROC
sensitivity
1 - specificity
0.0
0.2
0.4
0.6
0.8
1.0


precision-recall curve
precision
recall
1.0
0.9
0.8
0.7
0.6
0.5
0.0
0.2
0.4
0.6
0.8
1.0


20
15
10
5
0
−6
−4
−2
0
2
4
6


ROC
sensitivity
1 - specificity
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

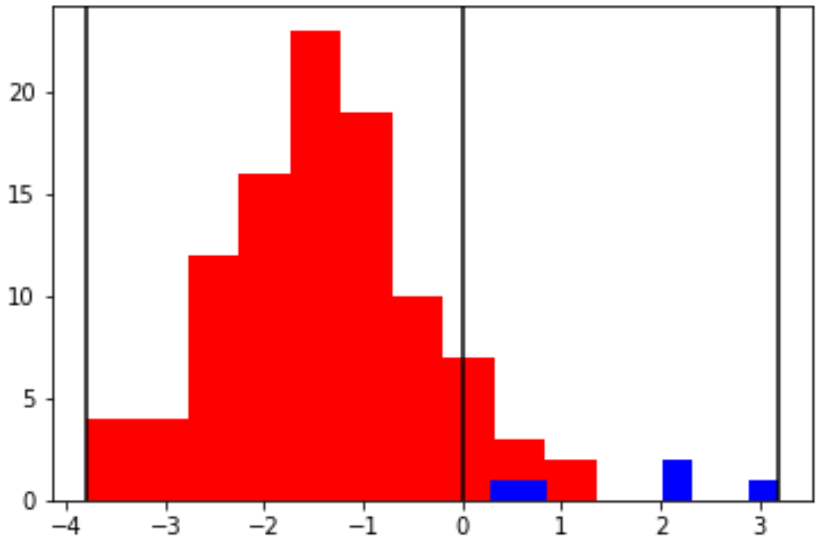

precision-recall curve
precision
recall
1.0
0.9
0.8
0.7
0.6
0.5
0.0
0.2
0.4
0.6
0.8
1.0

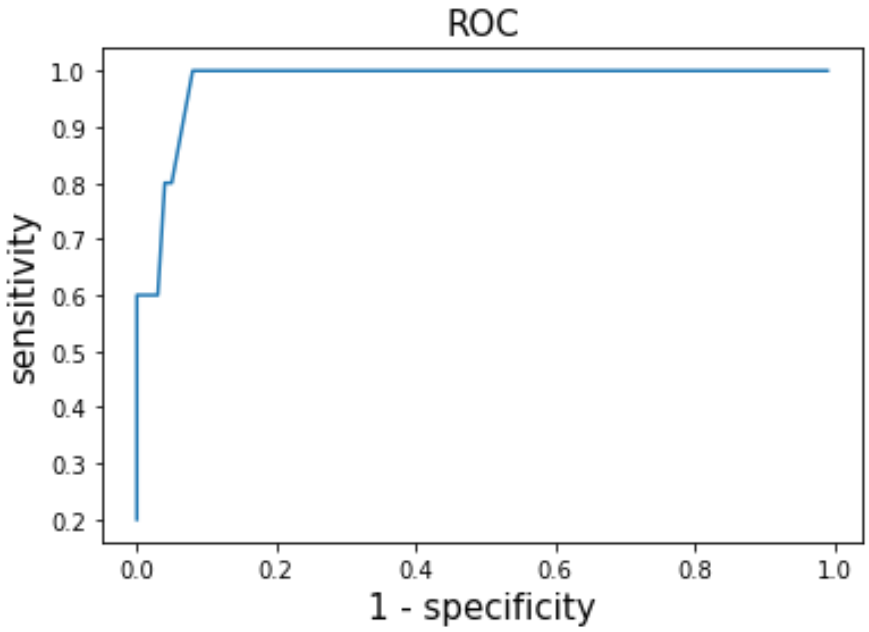
As mu gets larger, ROC and precision-recall curves get closer and closer to 1. This means we are making better and better predictions.

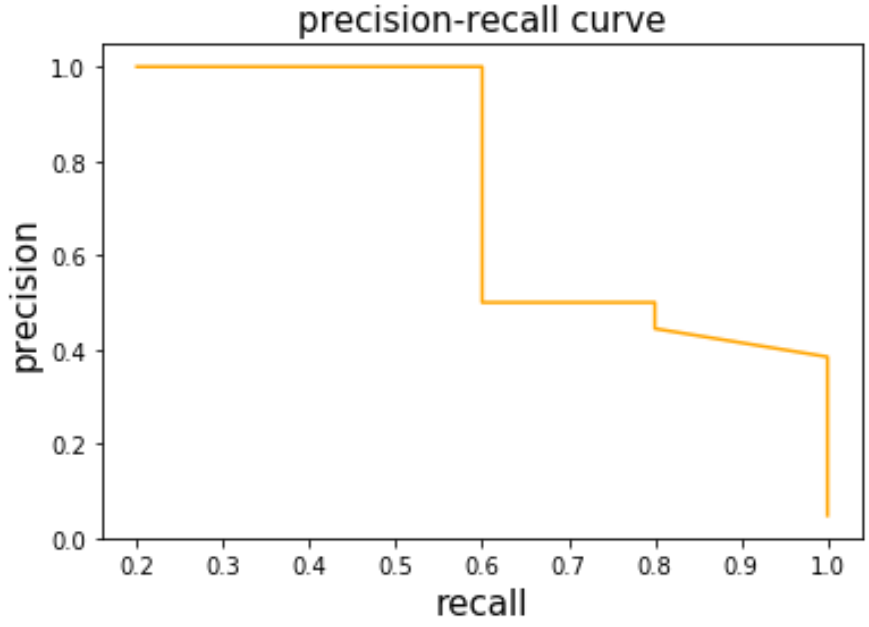
Question 4c

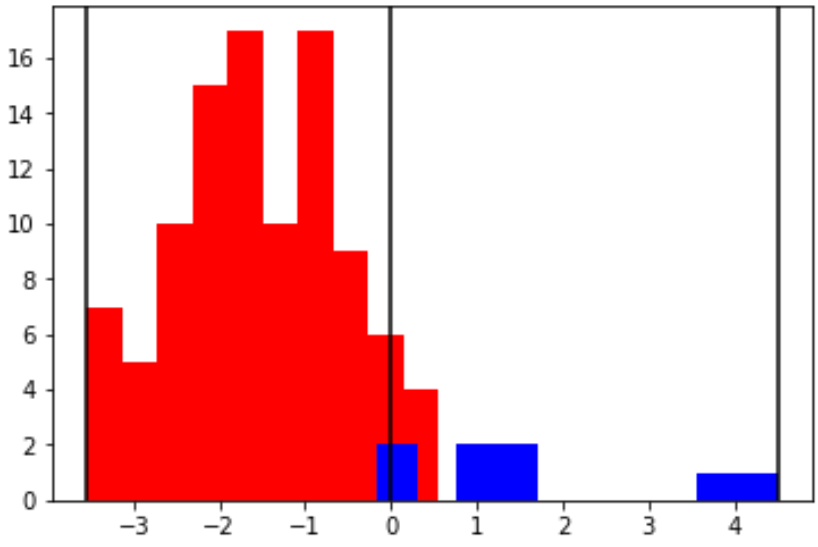
In [16]:

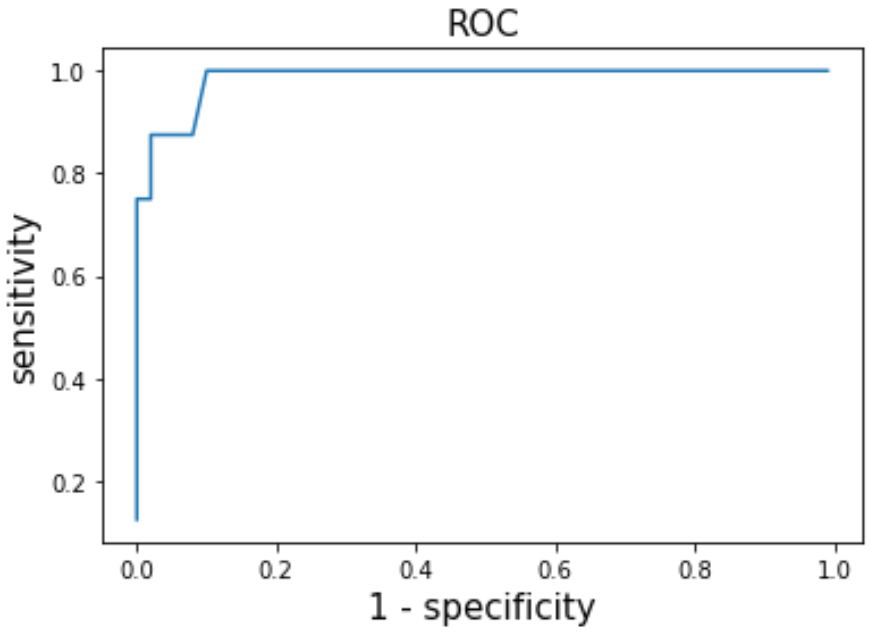
```
Class1_observations=(5,8,23,34,49,66,88,93,100)

for i in range(9):
    X_class0 = np.random.normal(-1.5,1,100)
    X_class1 = np.random.normal(1.5,1,Class1_observations[i])

    one_minus_specificity=[]
    sensitivity=[]
    precision=[]
    cutoff=np.linspace(min(min(X_class0),min(X_class1)),max(max(X_class1),max(X_
class0))-0.000000001,70)

    plt.hist(X_class0, color='red')
    plt.hist(X_class1, color='blue')
    plt.axvline(x=0, color='black')
    plt.axvline(x=min(min(X_class0),min(X_class1)), color='black')
    plt.axvline(max(max(X_class1),max(X_class0))-0.000000001, color='black')
    plt.show()


    for j in range(len(cutoff)):
        TP=0
        FP=0
        FN=0
        TN=0
        TP += len(X_class1[X_class1 > cutoff[j]])
        FP += len(X_class0[X_class0 > cutoff[j]])
        FN += len(X_class1[X_class1 <= cutoff[j]])
        TN += len(X_class0[X_class0 <= cutoff[j]])
        N=FP+TN
        P=TP+FN

        one_minus_specificity.append(1-(TN/N))
        sensitivity.append(TP/P)
        precision.append(TP/(TP+FP))

    plt.ylabel('sensitivity', fontsize = 15)
    plt.xlabel('1 - specificity', fontsize=15)
    plt.title("ROC", fontsize=15)
    plt.plot(one_minus_specificity,sensitivity)
    plt.show()

    plt.ylabel('precision', fontsize = 15)
    plt.xlabel('recall', fontsize=15)
    plt.title("precision-recall curve", fontsize=15)
    plt.plot(sensitivity,precision, color='orange')
    plt.show()
```

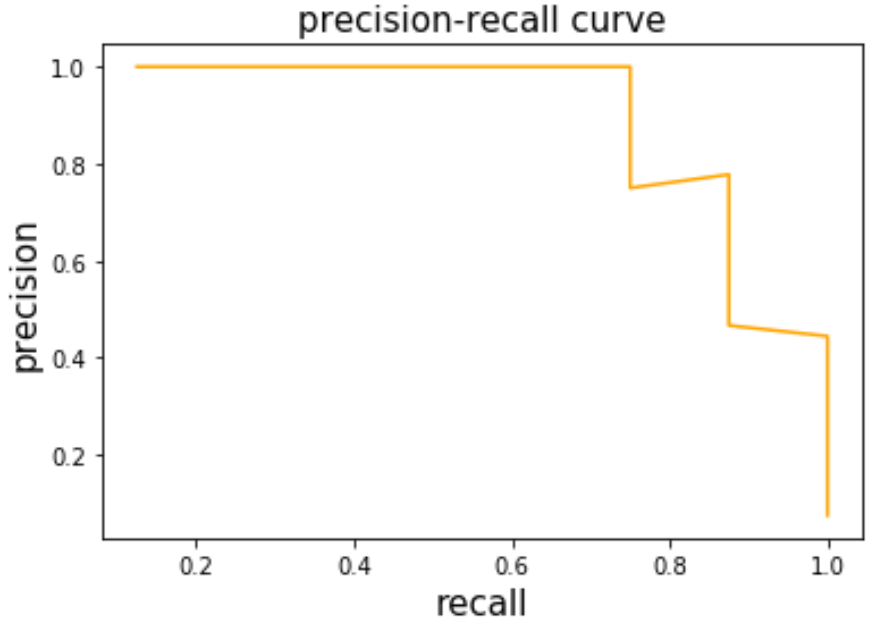

20
15
10
5
0
−4
−3
−2
−1
0
1
2
3

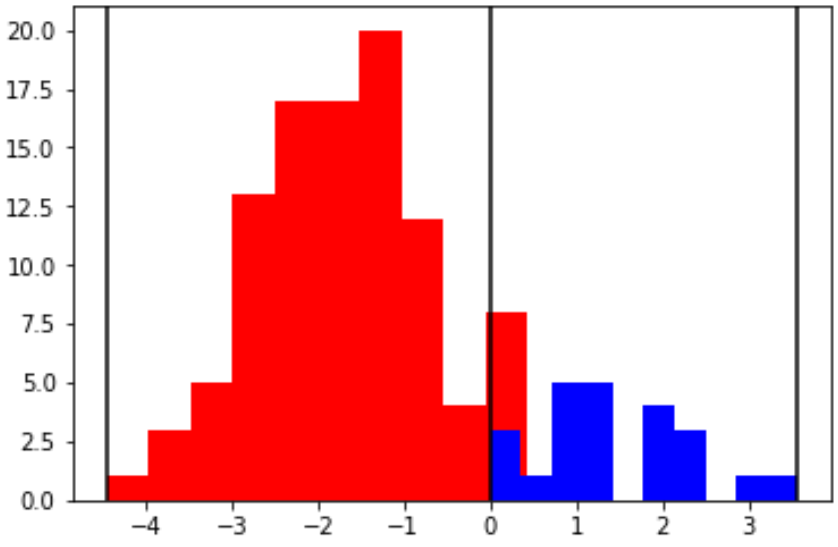

ROC
sensitivity
1 - specificity
0.0
0.2
0.4
0.6
0.8
1.0

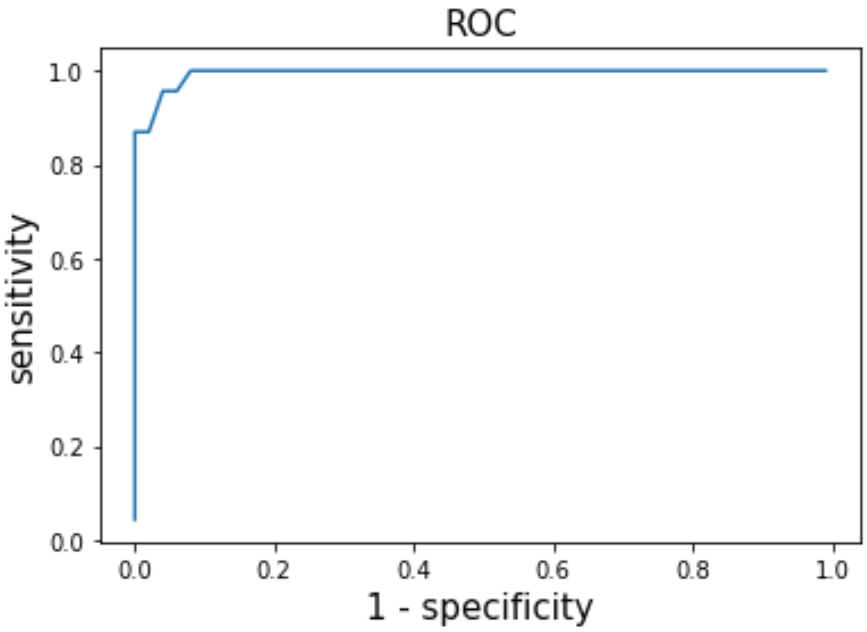

precision-recall curve
precision
recall
0.0
0.2
0.4
0.6
0.8
1.0

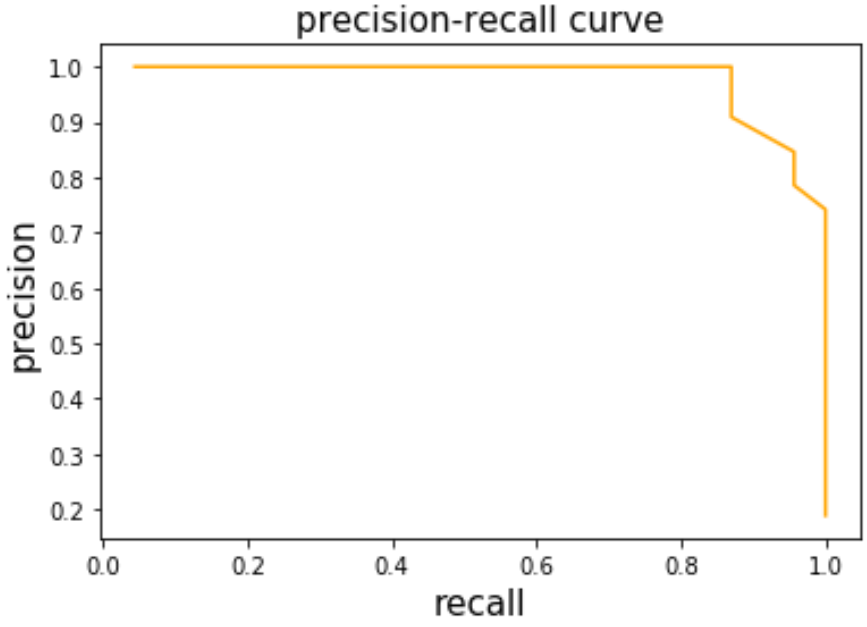

16
14
12
10
8
6
4
2
0
−3
−2
−1
0
1
2
3
4

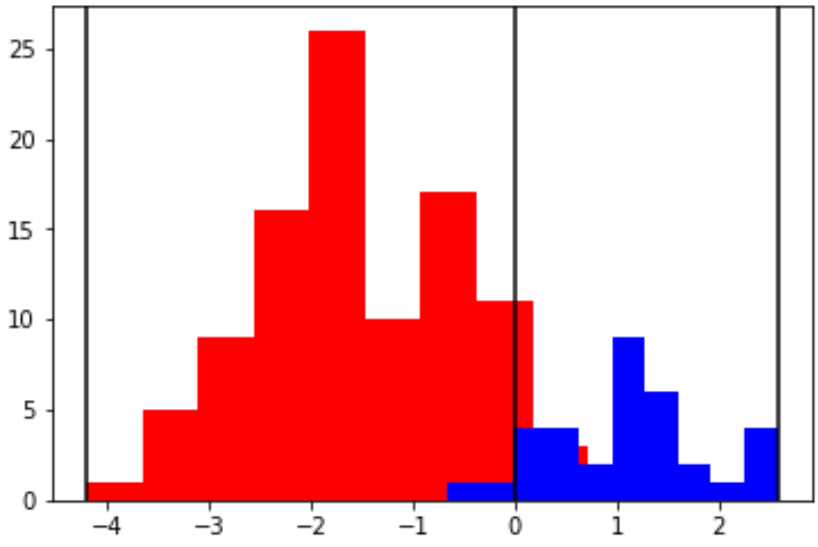

ROC
sensitivity
1 - specificity
1.0
0.8
0.6
0.4
0.2
0.0
0.2
0.4
0.6
0.8
1.0

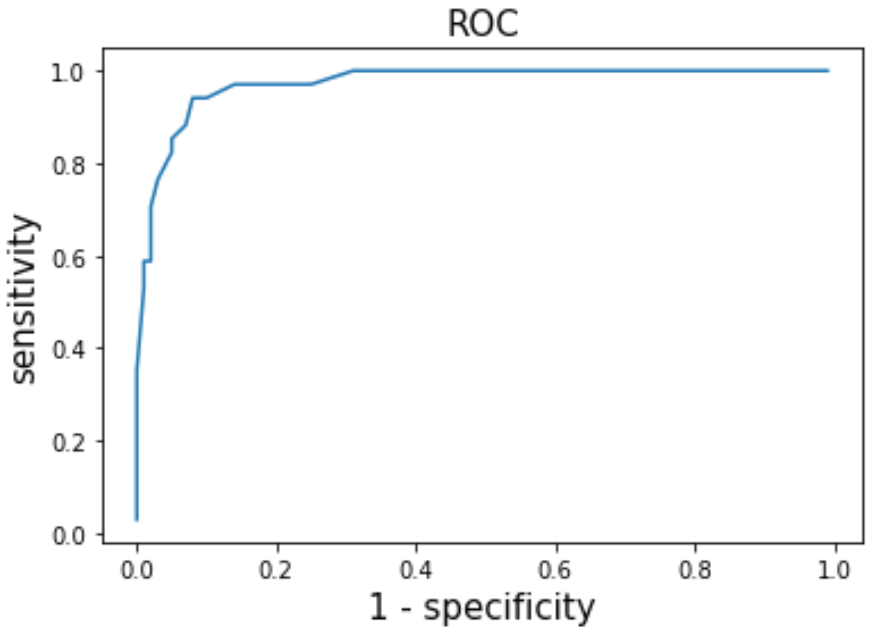

precision-recall curve
precision
recall
1.0
0.8
0.6
0.4
0.2
0.2
0.4
0.6
0.8
1.0

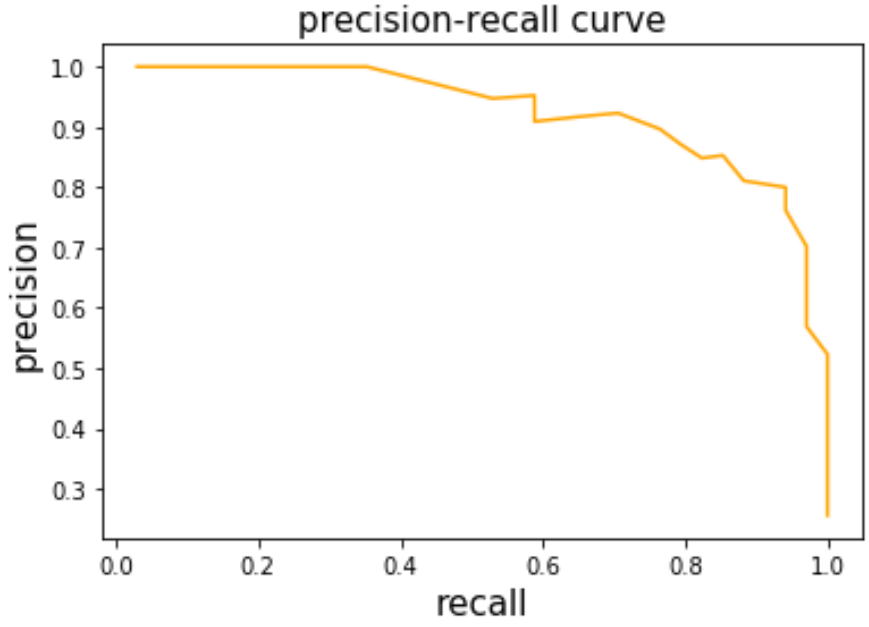

20.0
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
−4
−3
−2
−1
0
1
2
3

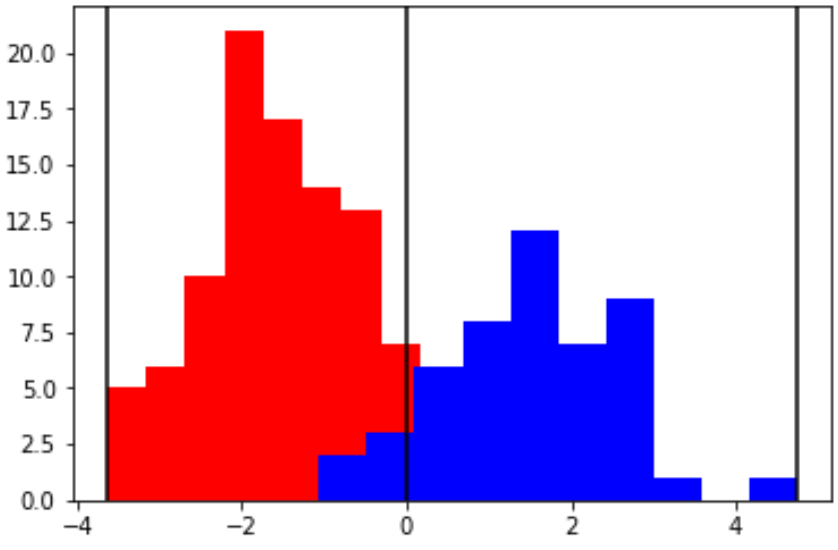

ROC
sensitivity
1 - specificity
0.0
0.2
0.4
0.6
0.8
1.0

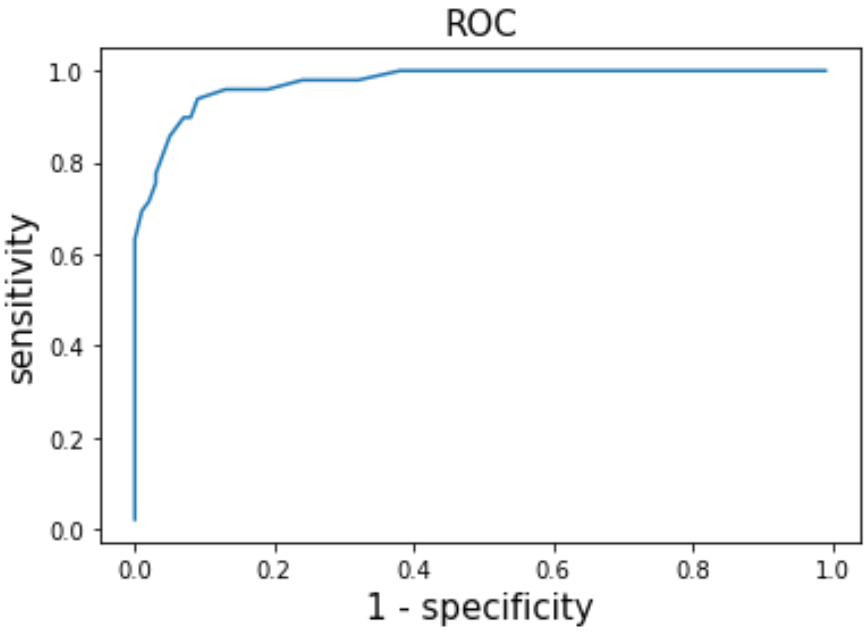

precision-recall curve
precision
recall
0.0
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1.0

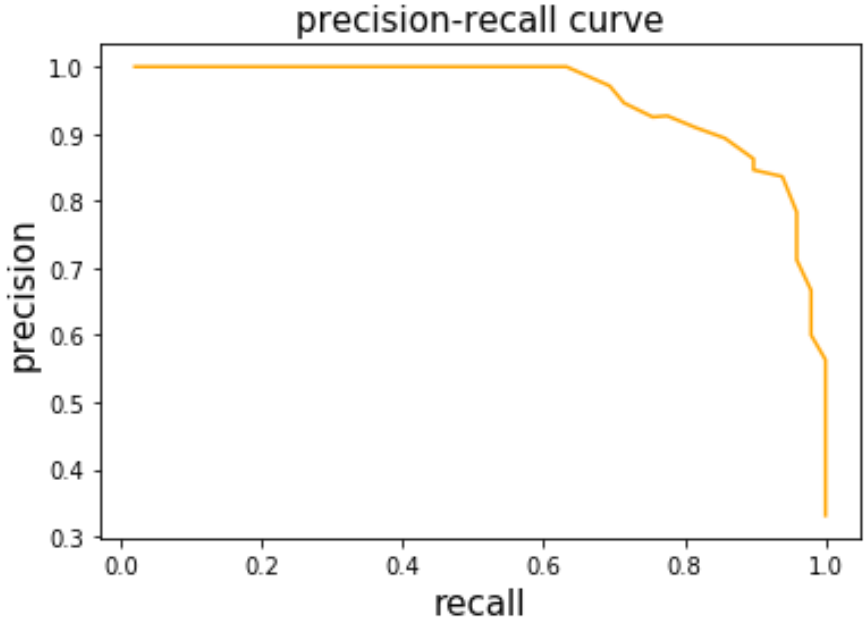

25
20
15
10
5
0
−4
−3
−2
−1
0
1
2

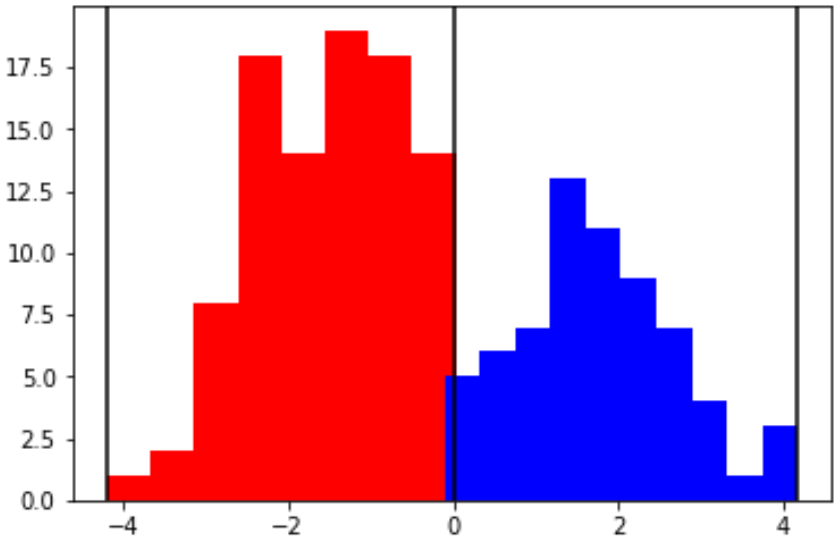

ROC
sensitivity
1 - specificity
0.0
0.2
0.4
0.6
0.8
1.0

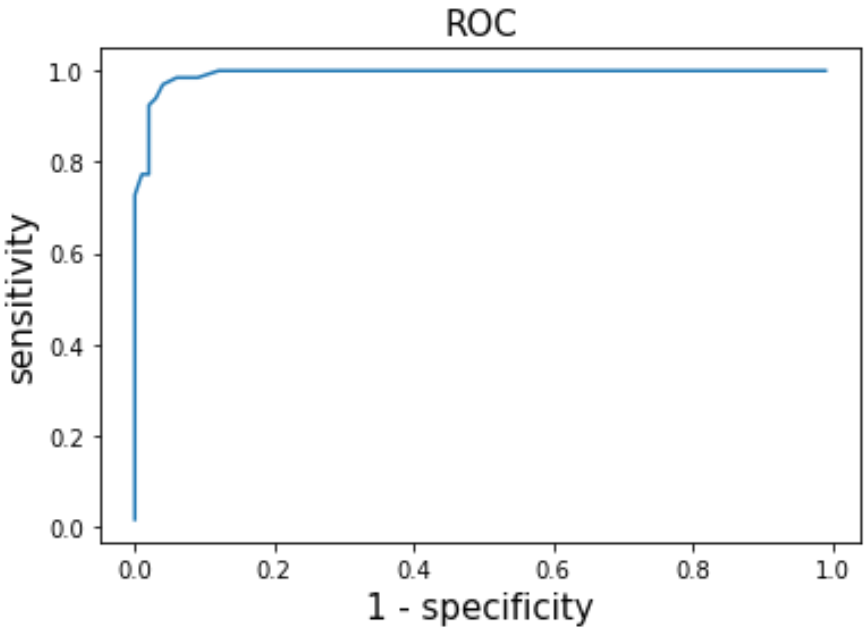

precision-recall curve
precision
recall
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.0
0.2
0.4
0.6
0.8
1.0

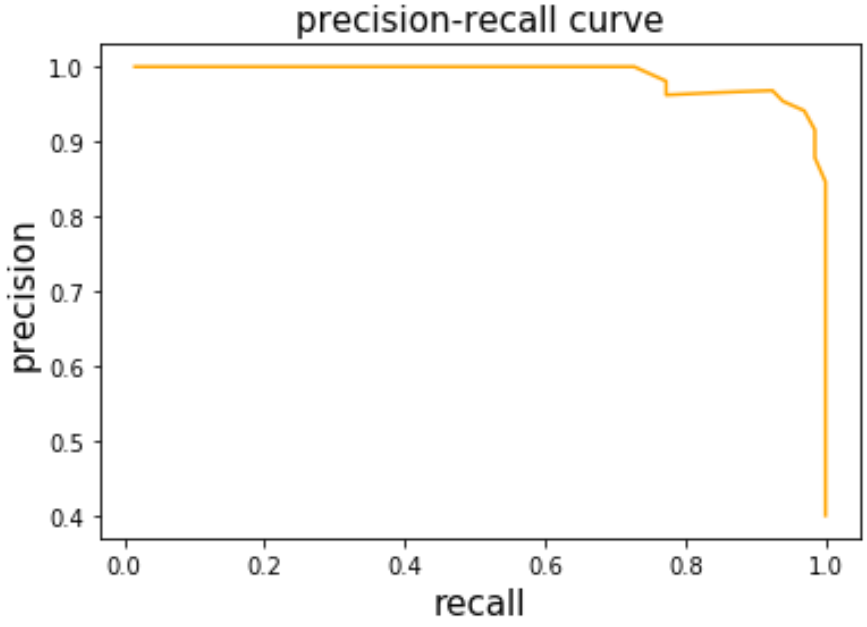

20.0
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
−4
−2
0
2
4

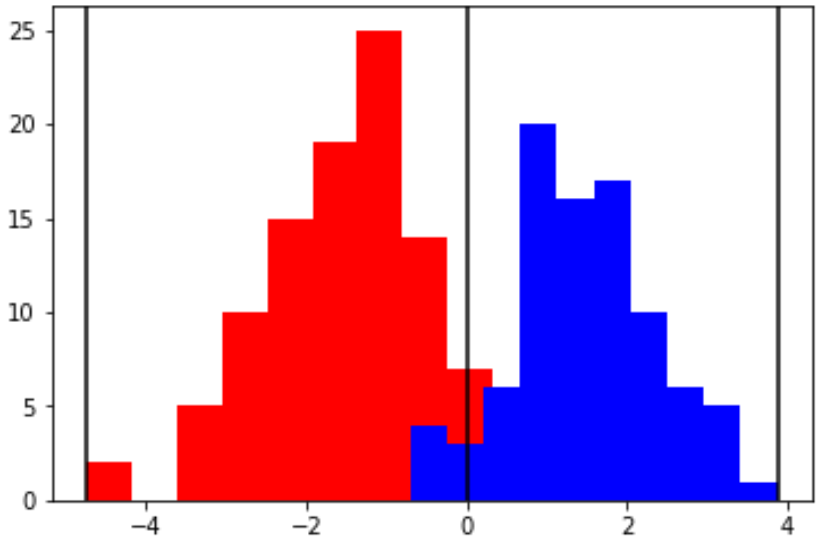

ROC
sensitivity
1 - specificity
0.0
0.2
0.4
0.6
0.8
1.0

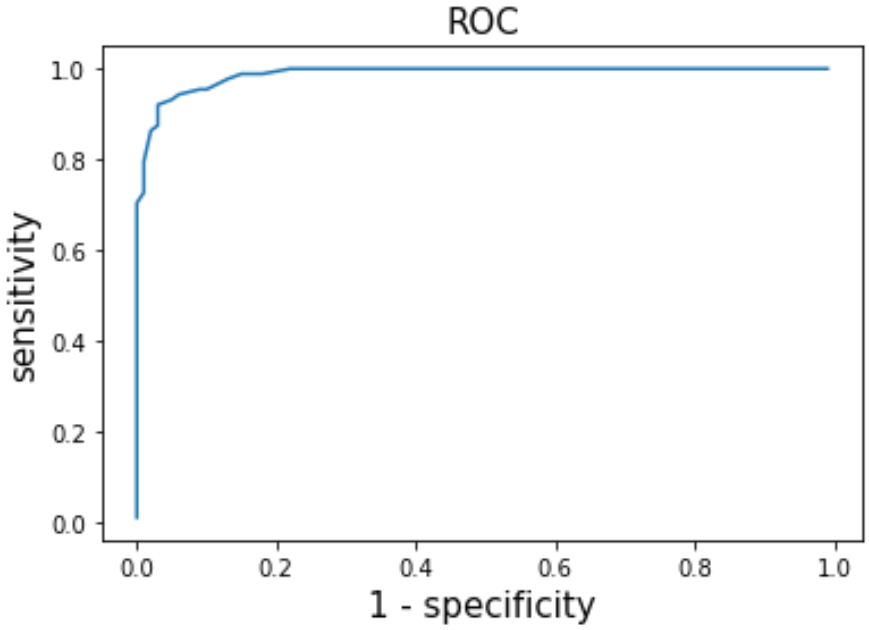

precision-recall curve
precision
recall
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.0
0.2
0.4
0.6
0.8
1.0

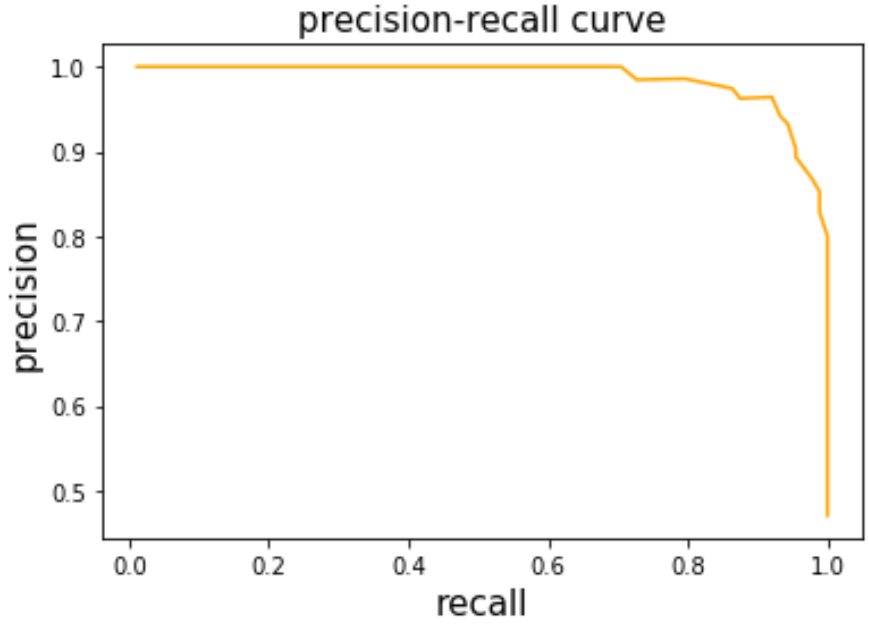

17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
−4
−2
0
2
4

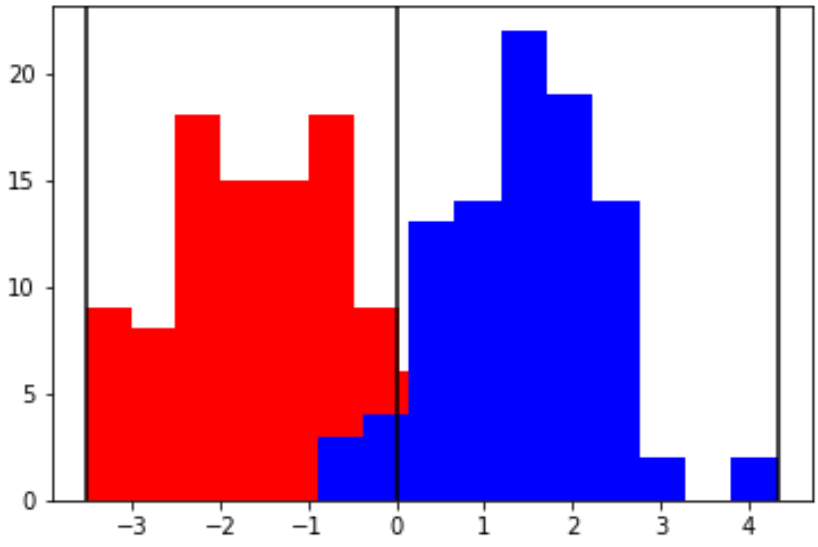

ROC
sensitivity
1 - specificity
0.0
0.2
0.4
0.6
0.8
1.0

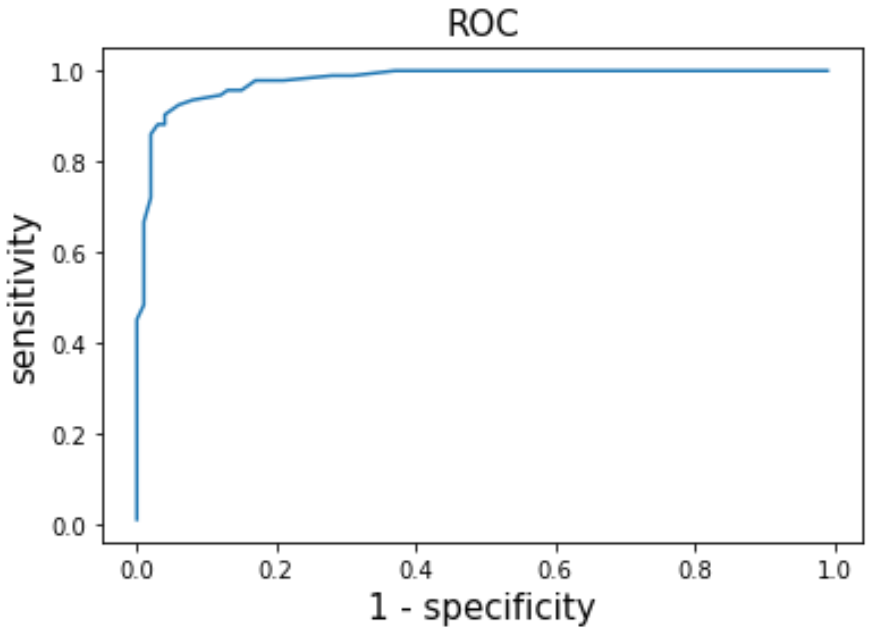

precision-recall curve
precision
recall
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.0
0.2
0.4
0.6
0.8
1.0

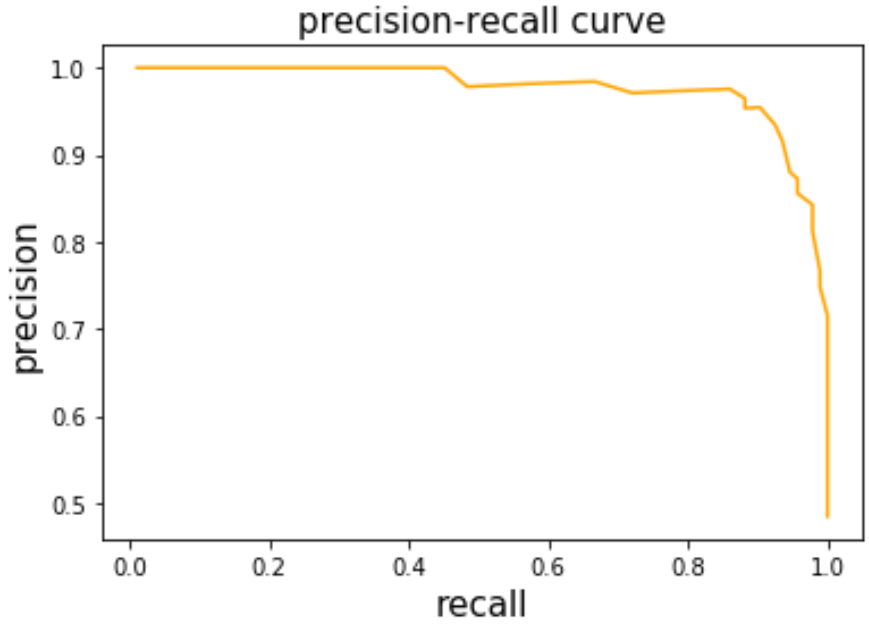

25
20
15
10
5
0
−4
−2
0
2
4

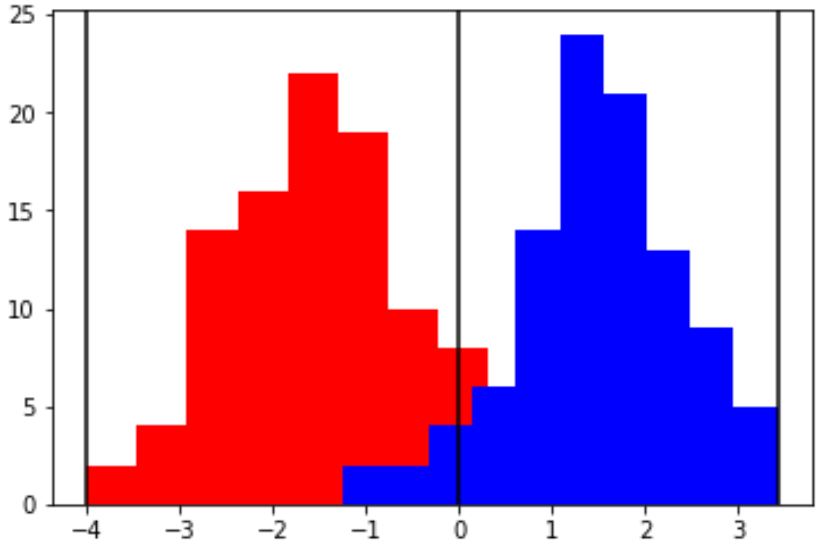

ROC
sensitivity
1 - specificity
0.0
0.2
0.4
0.6
0.8
1.0

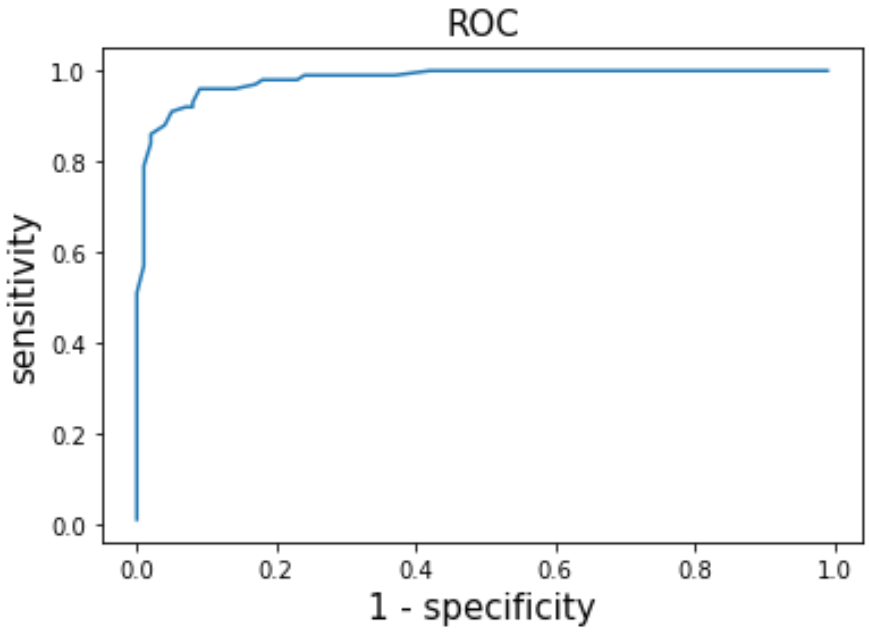

precision-recall curve
precision
recall
1.0
0.9
0.8
0.7
0.6
0.5
0.0
0.2
0.4
0.6
0.8
1.0

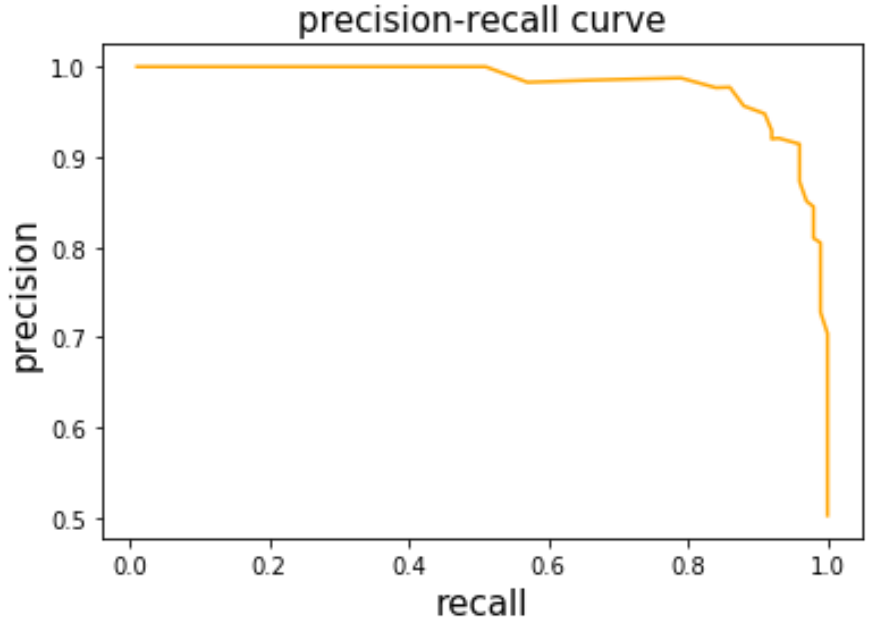

20
15
10
5
0
−3
−2
−1
0
1
2
3
4


ROC
sensitivity
1 - specificity
0.0
0.2
0.4
0.6
0.8
1.0


precision-recall curve
precision
recall
0.5
0.6
0.7
0.8
0.9
1.0
0.0
0.2
0.4
0.6
0.8
1.0


25
20
15
10
5
0
−4
−3
−2
−1
0
1
2
3


ROC
sensitivity
1 - specificity
0.0
0.2
0.4
0.6
0.8
1.0

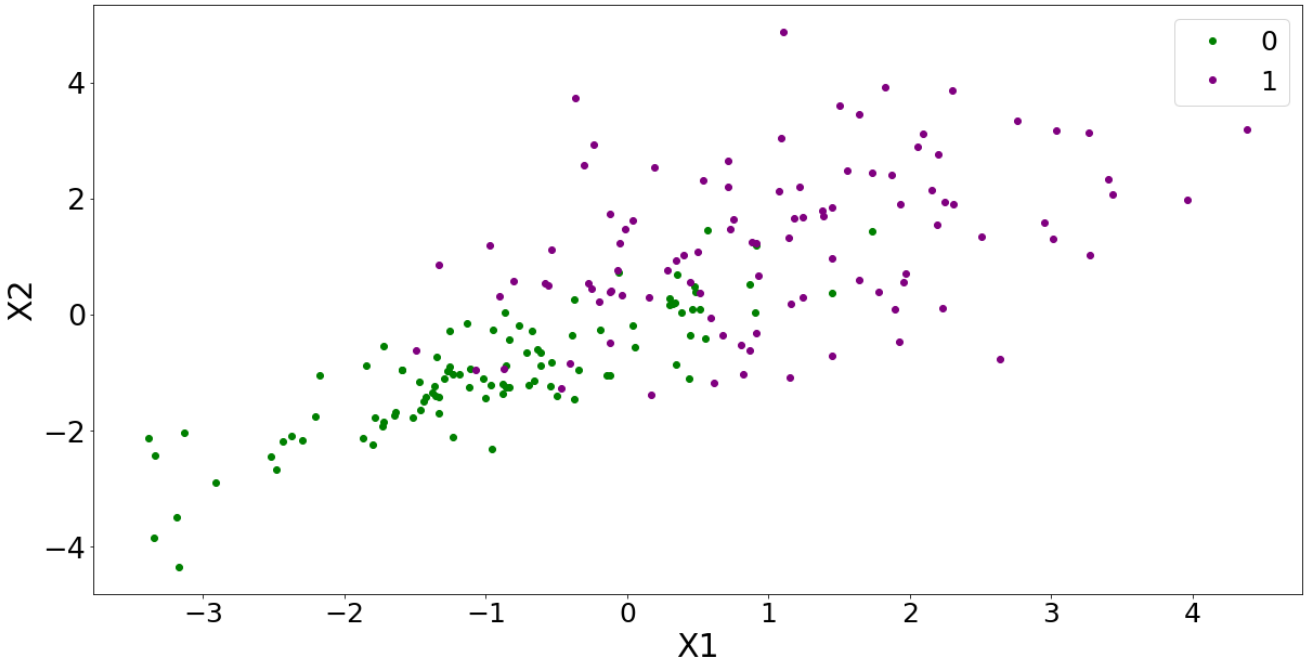

precision-recall curve
precision
recall
1.0
0.9
0.8
0.7
0.6
0.5
0.0
0.2
0.4
0.6
0.8
1.0

As class imbalance gets larger, ROC and precision-recall curves get worse. This means we need a good balance between classes to make good predictions.

Q6 a)

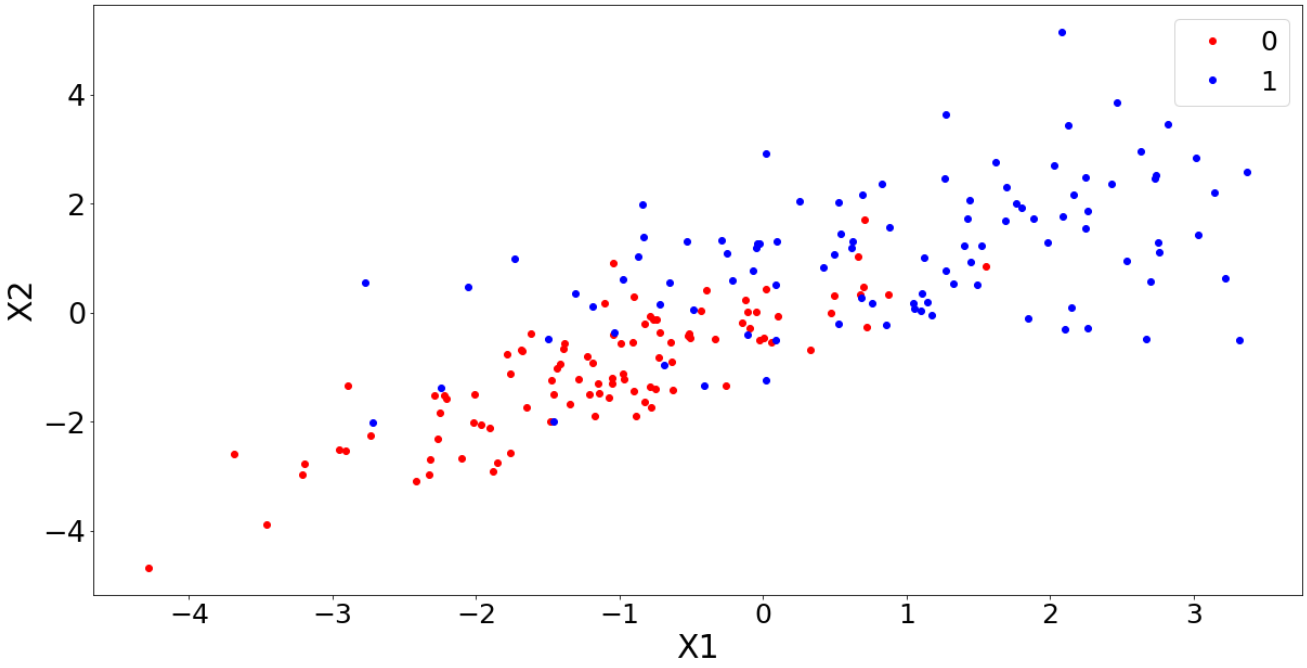
In [65]:

```
fig=plt.figure(figsize=(20,10))

y_class0_train=[]
y_class1_train=[]
for i in range(100):
    y_class0_train.append(0)
    y_class1_train.append(1)


mean_class0_train = [-1, -1]
cov_class0_train = [[1, 0.8], [0.8, 1]]
x1_class0_train, x2_class0_train = np.random.multivariate_normal(mean_class0_tra
in, cov_class0_train, 100).T
plt.plot(x1_class0_train, x2_class0_train, 'o', color='green')


mean_class1_train = [1, 1]
cov_class1_train = [[1.5, 0.675], [0.675, 1.5]]
x1_class1_train, x2_class1_train = np.random.multivariate_normal(mean_class1_tra
in, cov_class1_train, 100).T
plt.plot(x1_class1_train, x2_class1_train, 'o', color='purple')


plt.yticks(fontsize =27)
plt.xticks(fontsize =25)
plt.xlabel('X1', fontsize=30)
plt.ylabel('X2', fontsize=30)
plt.legend(["0", "1"],fontsize=25)

plt.show()
```


0
1
X2
X1
4
2
0
−2
−4
−3
−2
−1
0
1
2
3
4

In [66]:

```
fig=plt.figure(figsize=(20,10))

y_class0_valid=[]
y_class1_valid=[]
for i in range(100):
    y_class0_valid.append(0)
    y_class1_valid.append(1)

mean_class0_valid = [-1, -1]
cov_class0_valid = [[1, 0.8], [0.8, 1]]
x1_class0_valid, x2_class0_valid = np.random.multivariate_normal(mean_class0_val
id, cov_class0_valid, 100).T
plt.plot(x1_class0_valid, x2_class0_valid, 'o', color='red')


mean_class1_valid = [1, 1]
cov_class1_valid = [[1.5, 0.675], [0.675, 1.5]]
x1_class1_valid, x2_class1_valid = np.random.multivariate_normal(mean_class1_val
id, cov_class1_valid, 100).T
plt.plot(x1_class1_valid, x2_class1_valid, 'o', color='blue')


plt.yticks(fontsize =27)
plt.xticks(fontsize =25)
plt.xlabel('X1', fontsize=30)
plt.ylabel('X2', fontsize=30)
plt.legend(["0", "1"],fontsize=25)

plt.show()
```



Question 6b

In [594]:

```
def sigmoid(z): #sigmoid for logistic regression
    return 1 / (1+np.exp(-z))

def gradient_descent(x, y, initial_theta, loss_func, alpha=0.01, precision=0.001
):
    loss = []
    theta = initial_theta
    all_thetas1 = [] # to store all thetas
    predictions = [] # to store all predictions
    number_of_steps = 0
    previous_loss = 0
    prediction = sigmoid(np.dot(x,theta)) #dot product
    error = y - prediction
    current_loss = loss_func(y,prediction)/len(y)
    predictions.append(prediction)
    loss.append(current_loss)
    all_thetas1.append(theta)
    number_of_steps+=1
    while abs(current_loss - previous_loss) > precision: #if the difference betw
een current and previous values of loss function is bigger than the precision we
set
        previous_loss = current_loss #we update the value of the loss function t
o be the current value of loss function
        gradient = np.dot(error,x) #new gradient
        theta = theta + alpha * gradient  #update theta
        all_thetas1.append(theta)

        prediction = sigmoid(np.dot(x,theta))
        error = y - prediction
        current_loss = loss_func(y,prediction)/len(y)
        loss.append(current_loss)
        predictions.append(prediction)

    return all_thetas1, loss, predictions

def loss_func(y,prediction): #loss function for logistic regression
    return np.sum(-(y * np.log(np.abs(prediction)) + (1-y) * np.log(1.0000000001
-np.abs(prediction))))
```

In [595]:

```
initial_theta=np.random.rand(3)
X1_6b=np.concatenate((x1_class0_train,x1_class1_train))
X2_6b=np.concatenate((x2_class0_train,x2_class1_train))
Y1_6b_train=np.concatenate((y_class0_train,y_class1_train))
matrix_x_6b = np.c_[np.ones(X1_6b.shape[0]), X1_6b, X2_6b]
```

In [596]:

```
all_thetas1, loss, predictions = gradient_descent(matrix_x_6b,Y1_6b_train,initia
l_theta,loss_func, alpha=0.01, precision=0.001)
theta_6b = all_thetas1[-1]
decision_boundary=sigmoid(np.dot(matrix_x_6b,theta_6b))
print(theta_6b)
```

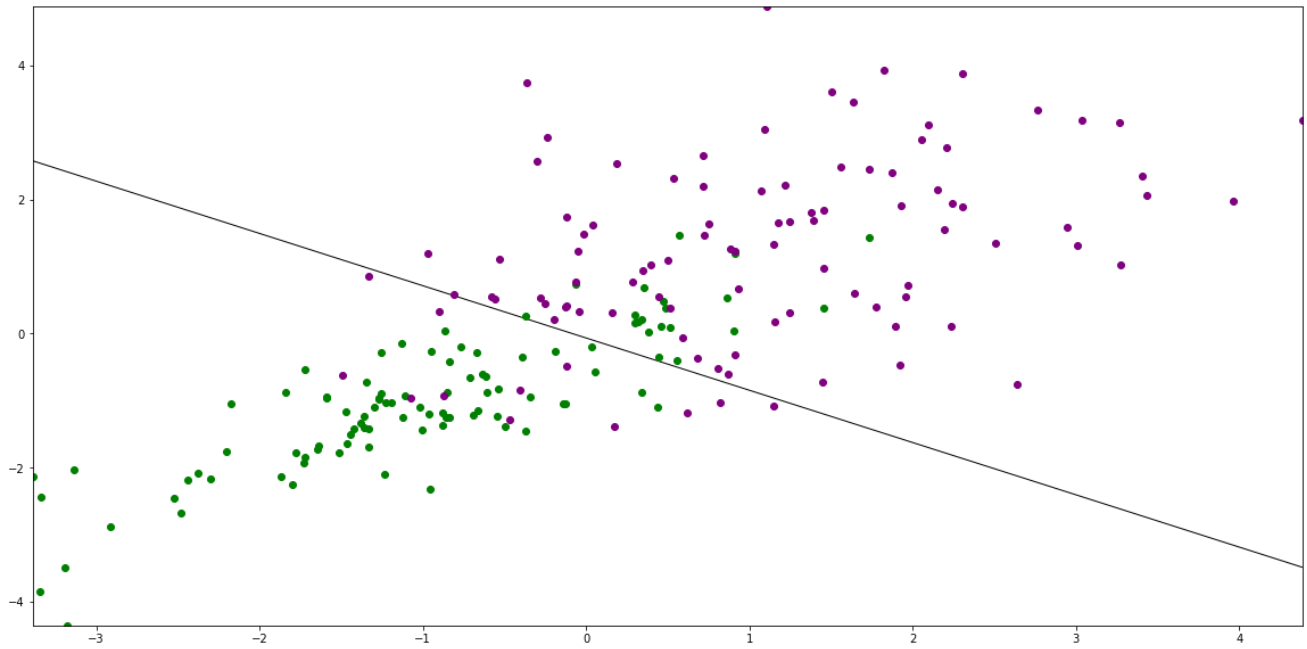
```
[ 0.07430534  0.87676614  1.12441629]
```

In [767]:

```
xx1, xx2 = np.meshgrid(np.linspace(np.min(X1_6b), np.max(X1_6b)), np.linspace(np
.min(X2_6b), np.max(X2_6b)))
grid = np.c_[np.ones(xx1.ravel().shape[0]), xx1.ravel(), xx2.ravel()]
probs = sigmoid(np.dot(grid,theta_6b)).reshape(xx1.shape)
```

In [768]:

```
fig=plt.figure(figsize=(20,10))
plt.plot(x1_class0_train, x2_class0_train, 'o', color='green')
plt.plot(x1_class1_train, x2_class1_train, 'o', color='purple')
plt.contour(xx1, xx2, probs, [0.5], linewidths=1, colors='black');
plt.show() #PLOTTING THE DECISION BOUNDARY for 6b
```



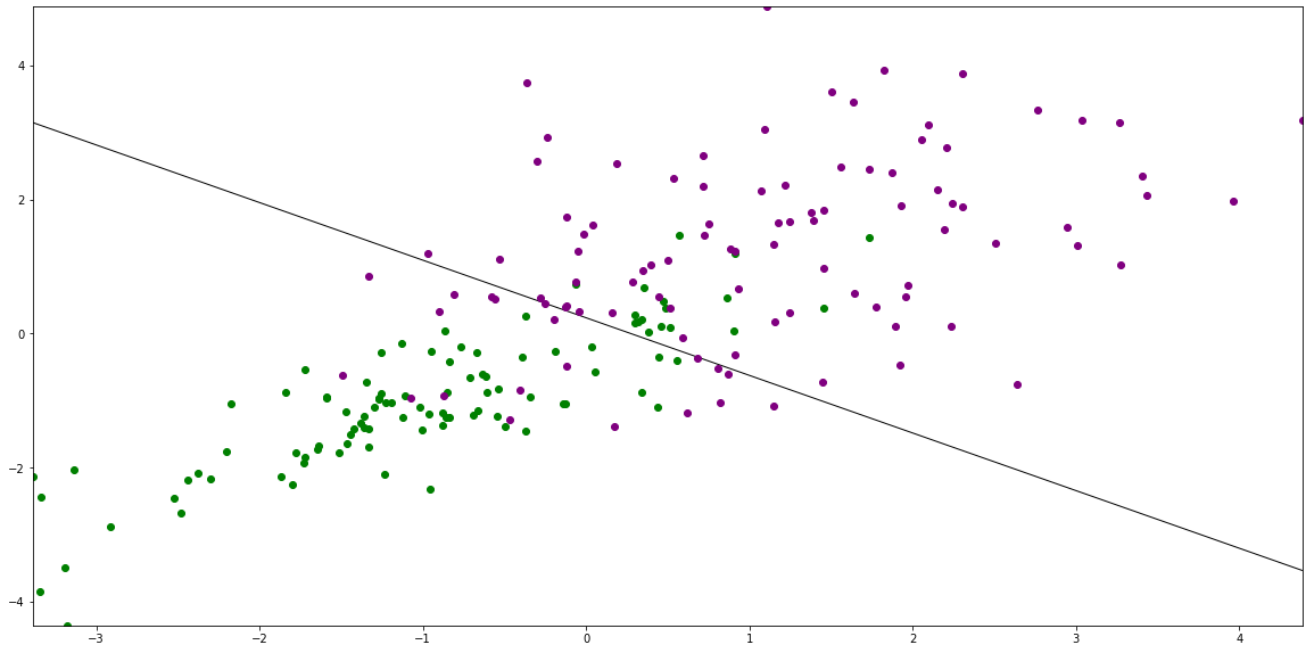
6c

In [765]:

```
xx1c, xx2c, xx3c, xx4c = np.meshgrid(np.linspace(np.min(X1_6b), np.max(X1_6b)),
np.linspace(np.min(X2_6b), np.max(X2_6b)),np.linspace(np.min(np.square(X1_6b)),
np.max(np.square(X1_6b))),np.linspace(np.min(np.square(X2_6b)), np.max(np.square
(X2_6b))))
grid = np.c_[np.ones(xx1c.ravel().shape[0]), xx1c.ravel(), xx2c.ravel(), xx3c.ra
vel(), xx4c.ravel()]
probs_c = sigmoid(np.dot(grid,theta_6c)).reshape(xx1c.shape)
```

In [766]:

```
fig=plt.figure(figsize=(20,10))
plt.plot(x1_class0_train, x2_class0_train, 'o', color='green')
plt.plot(x1_class1_train, x2_class1_train, 'o', color='purple')
plt.contour(xx1c[:,:,0,0], xx2c[:,:,0,0], probs_c[:,:,0,0], [0.5], linewidths=1,
colors='black');
plt.show() #DECISION BOUNDARY for 6c
```



LDA

In [655]:

```
sum_over_x_class0_LDA=np.zeros((2,2))
for i in range(100):
    x_class0 = [x1_class0_train[i],x2_class0_train[i]]
    x_class0 = np.reshape(x_class0,(2,1))
    x_minus_mu = x_class0 - np.reshape([np.mean(x1_class0_train),np.mean(x2_clas
s0_train)],(2,1))
    sum_over_x_class0_LDA = sum_over_x_class0_LDA + (x_minus_mu*(x_minus_mu.T))
print(sum_over_x_class0_LDA)
```

```
[[ 118.31211982   95.09272211]
 [  95.09272211  106.21955506]]
```

In [656]:

```
sum_over_x_class1_LDA=np.zeros((2,2))
for i in range(100):
    x_class1 = [x1_class1_train[i],x2_class1_train[i]]
    x_class1 = np.reshape(x_class1,(2,1))
    x_minus_mu = x_class1 - np.reshape([np.mean(x1_class1_train),np.mean(x2_clas
s1_train)],(2,1))
    sum_over_x_class1_LDA = sum_over_x_class1_LDA + (x_minus_mu*(x_minus_mu.T))
print(sum_over_x_class1_LDA)
```

```
[[ 151.59543      68.02565726]
 [  68.02565726  185.0709931 ]]
```

In [657]:

```
var_cov_LDA = (sum_over_x_class0_LDA+sum_over_x_class1_LDA)/198
```

In [658]:

```
var_cov_LDA
```

Out[658]:

```
array([[ 1.36316944,  0.8238302 ],
       [ 0.8238302 ,  1.47116438]])
```

In [659]:

```
a=np.linalg.inv(var_cov_LDA)
```

In [660]:

```
b=[[-1.96874],
   [-2.22045]]
```

In [661]:

```
np.dot(np.dot([[2,-2]],a),b)*0.5
```

Out[661]:

```
array([[ 0.25466417]])
```

In [662]:

```
np.dot(a,b)
```

Out[662]:

```
array([[-0.80427081],
       [-1.05893497]])
```

In [663]:

```
np.mean(x1_class0_train),np.mean(x2_class0_train)
```

Out[663]:

```
(-0.94297441828116146, -0.99894842668788242)
```

In [664]:

```
np.mean(x1_class1_train),np.mean(x2_class1_train)
```

Out[664]:

```
(1.0257684318824545, 1.2214961526111292)
```

In [665]:

```
np.linalg.inv(var_cov_LDA)
```

Out[665]:

```
array([[ 1.10884811, -0.62093847],
       [-0.62093847,  1.02745001]])
```

decision boundary for LDA

For LDA, we need to solve this for decision boundary:

$$\log\frac{\pi_0}{\pi_1} - \frac{1}{2}(\mu_0+\mu_1)^T\Sigma^{-1}(\mu_0-\mu_1) + X^T\Sigma^{-1}(\mu_0-\mu_1) = 0$$

$$\mu_0 = \begin{pmatrix} -0.94297 \\ -0.99895 \end{pmatrix} \qquad \mu_1 = \begin{pmatrix} 1.02577 \\ 1.22150 \end{pmatrix}$$

$$\Sigma = \begin{pmatrix} 1.36317 & 0.82383 \\ 0.82383 & 1.47116 \end{pmatrix} \qquad \mu_0+\mu_1 = \begin{pmatrix} 0.0828 \\ 0.22255 \end{pmatrix}$$

$$\Sigma^{-1} = \begin{pmatrix} 1.10885 & -0.62094 \\ -0.62094 & 1.02745 \end{pmatrix} \qquad \mu_0-\mu_1 = \begin{pmatrix} -1.96874 \\ -2.22045 \end{pmatrix}$$

$$\log\left(\frac{0.5}{0.5}\right) - \frac{1}{2}\begin{pmatrix}0.0828, 0.22255\end{pmatrix}\begin{pmatrix}1.10885 & -0.62094 \\ -0.62094 & 1.02745\end{pmatrix}\begin{pmatrix}-1.96874 \\ -2.22045\end{pmatrix} + X^T\begin{pmatrix}1.10885 & -0.62094 \\ -0.62094 & 1.02745\end{pmatrix}\begin{pmatrix}-1.96874 \\ -2.22045\end{pmatrix} = 0$$

$$0.25466 + [X_1, X_2]\begin{bmatrix} -0.80427 \\ -1.05893 \end{bmatrix} = 0$$

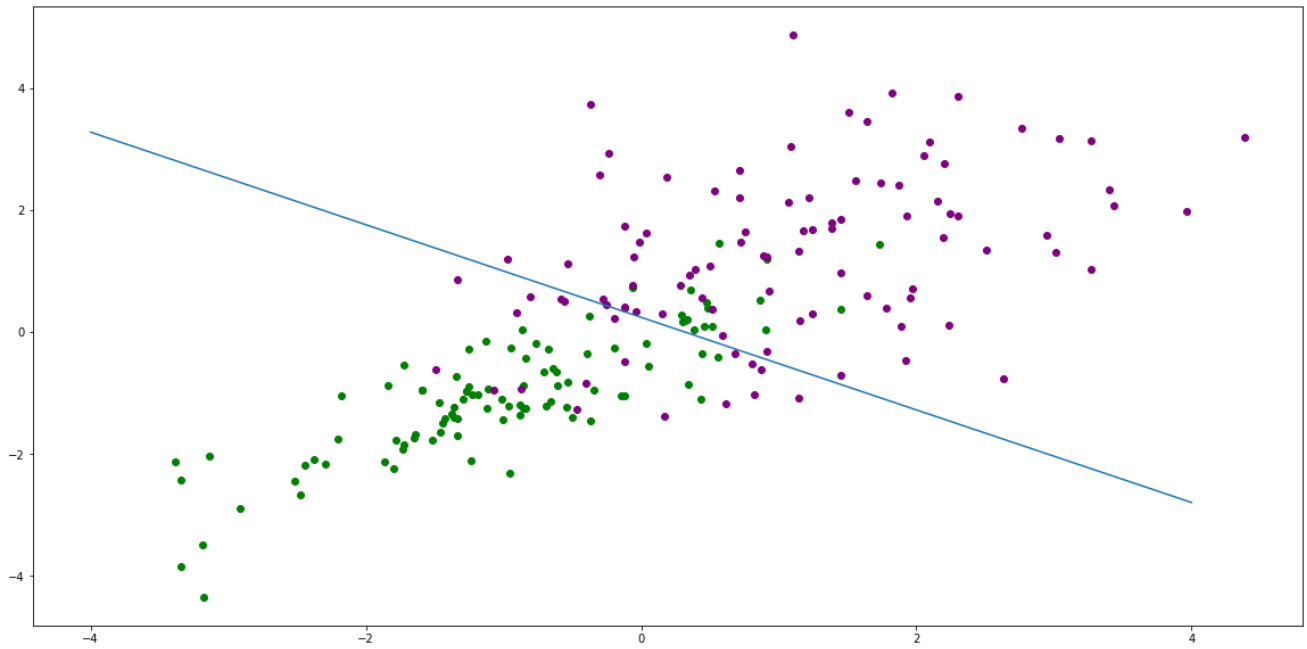
$$0.25466 - 0.80427 \cdot X_1 - 1.05893 \cdot X_2 = 0$$

$$1.05893 X_2 = -0.80427 X_1 + 0.25466$$

decision boundary for LDA $\Longrightarrow$ $X_2 = -0.75951 X_1 + 0.24049$

In [666]:

```
fig=plt.figure(figsize=(20,10))
plt.plot(x1_class0_train, x2_class0_train, 'o', color='green')
plt.plot(x1_class1_train, x2_class1_train, 'o', color='purple')
x = np.linspace(-4,4,1000000)
plt.plot(x, -0.75951*x+0.24049)
plt.show()
```



QDA

In [697]:

```
sum_over_x_class0_QDA=np.zeros((2,2))
for i in range(100):
    x_class0 = [x1_class0_train[i],x2_class0_train[i]]
    x_class0 = np.reshape(x_class0,(2,1))
    x_minus_mu = x_class0 - np.reshape([np.mean(x1_class0_train),np.mean(x2_clas
s0_train)],(2,1))
    sum_over_x_class0_LDA = sum_over_x_class0_LDA + (x_minus_mu*(x_minus_mu.T))
    var_cov_QDA_Class0 = (sum_over_x_class0_LDA)/99
print(var_cov_QDA_Class0)
```

```
[[ 3.58521575  2.88159764]
 [ 2.88159764  3.2187744 ]]
```

In [698]:

```
print(var_cov_QDA_Class0[0,0])
```

```
3.58521575198
```

$$\log\pi_0 - \frac{1}{2}\log(|\Sigma_0|) - \frac{1}{2}(\vec{x}-\vec{\mu}_0)^T\Sigma_0^{-1}(\vec{x}-\vec{\mu}_0)$$
$$= \log\pi_1 - \frac{1}{2}\log(|\Sigma_1|) - \frac{1}{2}(\vec{x}-\vec{\mu}_1)^T\Sigma_1^{-1}(\vec{x}-\vec{\mu}_1)$$

$$-\frac{1}{2}\log(|\Sigma_0|) - \frac{1}{2}(\vec{x}-\vec{\mu}_0)^T\Sigma_0^{-1}(\vec{x}-\vec{\mu}_0)$$
$$= -\frac{1}{2}\log(|\Sigma_1|) - \frac{1}{2}(\vec{x}-\vec{\mu}_1)^T\Sigma_1^{-1}(\vec{x}-\vec{\mu}_1)$$

$$(\vec{x}-\vec{\mu}_1)^T\Sigma_1^{-1}(\vec{x}-\vec{\mu}_1) - (\vec{x}-\vec{\mu}_0)^T\Sigma_0^{-1}(\vec{x}-\vec{\mu}_0) = \underbrace{\log|\Sigma_0| - \log|\Sigma_1|}_{\text{Set this as Constant } C}$$

$$(\vec{x}-\vec{\mu}_1)^T\Sigma_1^{-1}(\vec{x}-\vec{\mu}_1) - (\vec{x}-\vec{\mu}_0)^T\Sigma_0^{-1}(\vec{x}-\vec{\mu}_0) = C \quad (1)$$

let $x_0 = x - \mu_{00}$ $\quad x_1 = x - \mu_{10}$ $\quad \mu_0 = \begin{bmatrix}\mu_{00}\\ \mu_{01}\end{bmatrix}$

$y_0 = y - \mu_{01}$ $\quad y_1 = y - \mu_{11}$

$$\Sigma_1 = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \qquad \Sigma_2 = \begin{bmatrix} p & q \\ r & s \end{bmatrix} \qquad \mu_1 = \begin{bmatrix}\mu_{10}\\ \mu_{11}\end{bmatrix}$$

so, equation (1) becomes

$$\begin{bmatrix} x_1 & y_1 \end{bmatrix}\begin{bmatrix} a & b \\ c & d \end{bmatrix}\begin{bmatrix} x_1 \\ y_1 \end{bmatrix} - \begin{bmatrix} x_0 & y_0 \end{bmatrix}\begin{bmatrix} p & q \\ r & s \end{bmatrix}\begin{bmatrix} x_0 \\ y_0 \end{bmatrix} = C$$

$$\begin{bmatrix} x_1 & y_1 \end{bmatrix}\begin{bmatrix} ax_1+by_1 \\ cx_1+dy_1 \end{bmatrix} - \begin{bmatrix} x_0 & y_0 \end{bmatrix}\begin{bmatrix} px_0+qy_0 \\ rx_0+sy_0 \end{bmatrix} = C$$

$$x_1(ax_1+by_1) + y_1(cx_1+dy_1) - x_0(px_0+qy_0) - y_0(rx_0+sy_0) = C$$

$$ax_1^2 + bx_1y_1 + cx_1y_1 + dy_1^2 - px_0^2 - qx_0y_0 - rx_0y_0 - sy_0^2 = C$$

$$bx_1y_1 + cx_1y_1 + dy_1^2 - qx_0y_0 - rx_0y_0 - sy_0^2 = C - ax_1^2 + px_0^2$$

$$dy_1^2 - sy_0^2 + bx_1y_1 + cx_1y_1 - qx_0y_0 - rx_0y_0 = c - ax_1^2 + px_0^2$$

$$dy_1^2 - sy_0^2 + x_1y_1(b+c) + x_0y_0(-q-r) = c - ax_1^2 + px_0^2$$

substituting for x_0, y_0, x_1, y_1

$$d(y-\mu_{11})^2 - s(y-\mu_{01})^2 + (x-\mu_{10})(y-\mu_{11})(b+c) + (x-\mu_{00})(y-\mu_{01})(-q-r) = c - a(x-\mu_{10})^2 + p(x-\mu_{00})^2$$

$$d(y^2 - 2\mu_{11}y + \mu_{11}^2) - s(y^2 - 2\mu_{01}y + \mu_{01}^2) + (xy - x\mu_{11} - y\mu_{10} + \mu_{10}\mu_{11})(b+c) + (xy - x\mu_{01} - \mu_{00}y + \mu_{00}\mu_{01})(-q-r) = c - a(x-\mu_{10})^2 + p(x-\mu_{00})^2$$

$$dy^2 - 2\mu_{11}dy + d\mu_{11}^2 - sy^2 + 2s\mu_{01}y - s\mu_{01}^2 + xyb + xyc - x\mu_{11}b - x\mu_{11}c - y\mu_{10}b - y\mu_{10}c + \mu_{10}\mu_{11}b + \mu_{10}\mu_{11}c - xyq - xyr + x\mu_{01}q + x\mu_{01}r + \mu_{00}yq + \mu_{00}yr - \mu_{00}\mu_{01}q - \mu_{00}\mu_{01}r = c - a(x-\mu_{10})^2 + p(x-\mu_{00})^2$$

→ set as A → set as B

$$(d-s)y^2 + (-2\mu_{11}d + 2s\mu_{01} + xb + xc - \mu_{10}b - \mu_{10}c - xq - xr + \mu_{00}q + \mu_{00}r)y = c - a(x-\mu_{10})^2 + p(x-\mu_{00})^2 - d\mu_{11}^2 + s\mu_{01}^2 + x\mu_{11}b + x\mu_{11}c - \mu_{10}\mu_{11}b - \mu_{10}\mu_{11}c - x\mu_{01}q - x\mu_{01}r + \mu_{00}\mu_{01}q + \mu_{00}\mu_{01}r$$

set as −D.

in our question, we have $y \to x_2$. $x \to x_1$. $\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$

So, $Ax_2^2 + Bx_2 + D = 0$

So, $x_2 = \dfrac{-B \pm \sqrt{B^2 - 4AD}}{2A}$

In [699]:

```
sum_over_x_class1_QDA=np.zeros((2,2))
for i in range(100):
    x_class1 = [x1_class1_train[i],x2_class1_train[i]]
    x_class1 = np.reshape(x_class1,(2,1))
    x_minus_mu = x_class1 - np.reshape([np.mean(x1_class1_train),np.mean(x2_clas
s1_train)],(2,1))
    sum_over_x_class1_LDA = sum_over_x_class1_LDA + (x_minus_mu*(x_minus_mu.T))
    var_cov_QDA_Class1 = (sum_over_x_class1_LDA)/99
print(var_cov_QDA_Class1)
```

```
[[ 4.59380091  2.06138355]
 [ 2.06138355  5.60821191]]
```

In [700]:

```
print(np.mean(x1_class0_train),np.mean(x2_class0_train),np.mean(x1_class1_train)
,np.mean(x2_class1_train))
```

```
-0.942974418281 -0.998948426688 1.02576843188 1.22149615261
```

In [729]:

```
a_QDA=var_cov_QDA_Class0[0,0]
b_QDA=var_cov_QDA_Class0[0,1]
c_QDA=var_cov_QDA_Class0[1,0]
d_QDA=var_cov_QDA_Class0[1,1]
p_QDA=var_cov_QDA_Class1[0,0]
q_QDA=var_cov_QDA_Class1[0,1]
r_QDA=var_cov_QDA_Class1[1,0]
s_QDA=var_cov_QDA_Class1[1,1]
mu00_QDA=np.mean(x1_class0_train)
mu01_QDA=np.mean(x2_class0_train)
mu10_QDA=np.mean(x1_class1_train)
mu11_QDA=np.mean(x2_class1_train)
```

In [732]:

```
x1 = np.linspace(-3.5,3.5,1000)
A=d_QDA-s_QDA
def B(x):
    return -2*mu11_QDA*d_QDA+2*s_QDA*mu01_QDA+x*b_QDA+x*c_QDA-mu10_QDA*b_QDA-mu1
0_QDA*c_QDA-x*q_QDA-x*r_QDA+mu00_QDA*q_QDA+mu00_QDA*r_QDA
def D(x):
    return -((np.log(np.linalg.det(var_cov_QDA_Class0))-np.log(np.linalg.det(var
_cov_QDA_Class1))-a_QDA*(x-mu10_QDA)**2+p_QDA*(x-mu00_QDA)**2-d_QDA*(mu11_QDA**2
)+s_QDA*(mu01_QDA**2)+x*mu11_QDA*b_QDA+x*mu11_QDA*c_QDA-mu10_QDA*mu11_QDA*b_QDA-
mu10_QDA*mu11_QDA*b_QDA-mu10_QDA*mu11_QDA*c_QDA-x*mu01_QDA*q_QDA-x*mu01_QDA*r_QD
A+mu00_QDA*mu01_QDA*q_QDA+mu00_QDA*mu01_QDA*r_QDA))
```

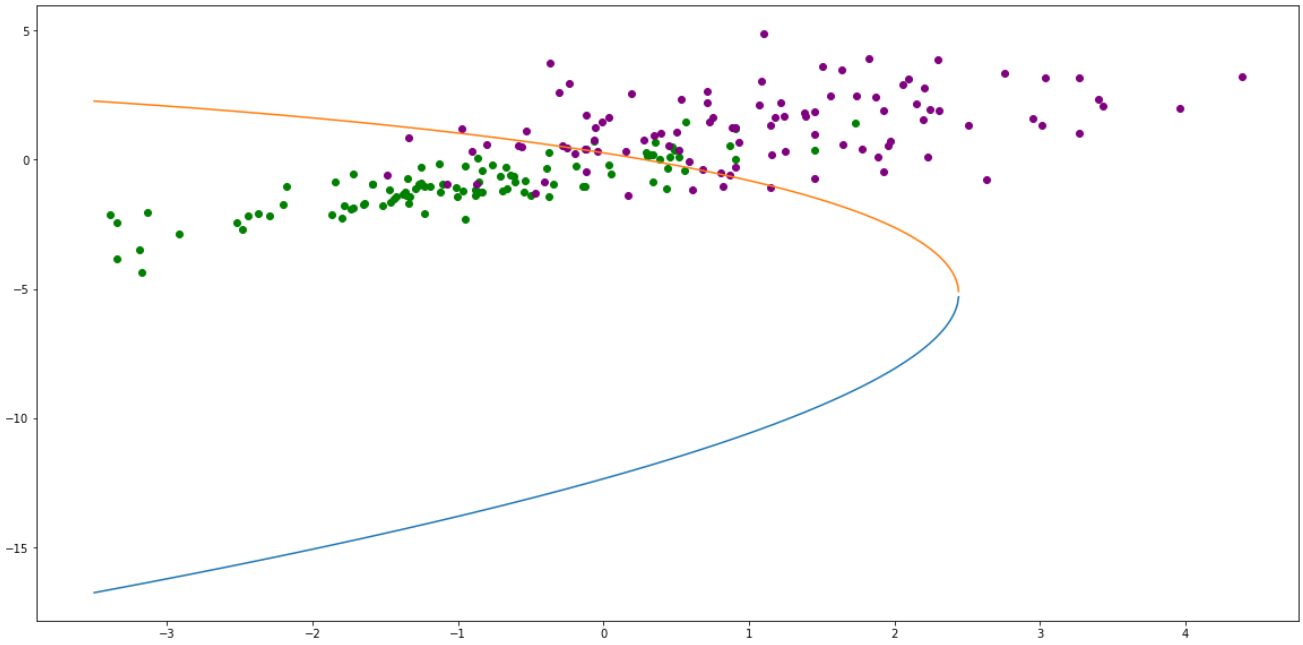
In [736]:

```
A=d_QDA-s_QDA
def B(x):
    return -2*mu11_QDA*d_QDA+2*s_QDA*mu01_QDA+x*b_QDA+x*c_QDA-mu10_QDA*b_QDA-mu1
0_QDA*c_QDA-x*q_QDA-x*r_QDA+mu00_QDA*q_QDA+mu00_QDA*r_QDA
def D(x):
    return -((np.log(np.linalg.det(var_cov_QDA_Class0))-np.log(np.linalg.det(var
_cov_QDA_Class1))-a_QDA*(x-mu10_QDA)**2+p_QDA*(x-mu00_QDA)**2-d_QDA*(mu11_QDA**2
)+s_QDA*(mu01_QDA**2)+x*mu11_QDA*b_QDA+x*mu11_QDA*c_QDA-mu10_QDA*mu11_QDA*b_QDA-
mu10_QDA*mu11_QDA*b_QDA-mu10_QDA*mu11_QDA*c_QDA-x*mu01_QDA*q_QDA-x*mu01_QDA*r_QD
A+mu00_QDA*mu01_QDA*q_QDA+mu00_QDA*mu01_QDA*r_QDA))
```

In [738]:

```
fig=plt.figure(figsize=(20,10))
plt.plot(x1_class0_train, x2_class0_train, 'o', color='green')
plt.plot(x1_class1_train, x2_class1_train, 'o', color='purple')
x1 = np.linspace(-3.5,3.5,1000)
plt.plot(x1, (-B(x1)+np.sqrt((B(x1)**2-4*A*D(x1))))/(2*A))
plt.plot(x1, (-B(x1)-np.sqrt((B(x1)**2-4*A*D(x1))))/(2*A))
plt.show()
```

```
/anaconda3/lib/python3.6/site-packages/ipykernel_launcher.py:5: Runt
imeWarning: invalid value encountered in sqrt
  """
/anaconda3/lib/python3.6/site-packages/ipykernel_launcher.py:6: Runt
imeWarning: invalid value encountered in sqrt
```



Naive Bayes

In [741]:

```
sum_over_x_class0_NB=np.zeros((2,2))
for i in range(100):
    x_class0 = [x1_class0_train[i],x2_class0_train[i]]
    x_class0 = np.reshape(x_class0,(2,1))
    x_minus_mu = x_class0 - np.reshape([np.mean(x1_class0_train),np.mean(x2_clas
s0_train)],(2,1))
    sum_over_x_class0_NB = sum_over_x_class0_NB + (x_minus_mu*(x_minus_mu.T))
    var_cov_NB_Class0 = (sum_over_x_class0_NB)/99
    var_cov_NB_Class0[1,0]=0
    var_cov_NB_Class0[0,1]=0
print(var_cov_NB_Class0)
```

```
[[ 1.19507192  0.        ]
 [ 0.          1.0729248 ]]
```

In [742]:

```
sum_over_x_class1_NB=np.zeros((2,2))
for i in range(100):
    x_class1 = [x1_class1_train[i],x2_class1_train[i]]
    x_class1 = np.reshape(x_class1,(2,1))
    x_minus_mu = x_class1 - np.reshape([np.mean(x1_class1_train),np.mean(x2_clas
s1_train)],(2,1))
    sum_over_x_class1_NB = sum_over_x_class1_NB + (x_minus_mu*(x_minus_mu.T))
    var_cov_NB_Class1 = (sum_over_x_class1_NB)/99
    var_cov_NB_Class1[1,0]=0
    var_cov_NB_Class1[0,1]=0
print(var_cov_NB_Class1)
```

```
[[ 1.53126697  0.        ]
 [ 0.          1.86940397]]
```

In [743]:

```
a_NB=var_cov_NB_Class0[0,0]
b_NB=var_cov_NB_Class0[0,1]
c_NB=var_cov_NB_Class0[1,0]
d_NB=var_cov_NB_Class0[1,1]
p_NB=var_cov_NB_Class1[0,0]
q_NB=var_cov_NB_Class1[0,1]
r_NB=var_cov_NB_Class1[1,0]
s_NB=var_cov_NB_Class1[1,1]
mu00_NB=np.mean(x1_class0_train)
mu01_NB=np.mean(x2_class0_train)
mu10_NB=np.mean(x1_class1_train)
mu11_NB=np.mean(x2_class1_train)
```

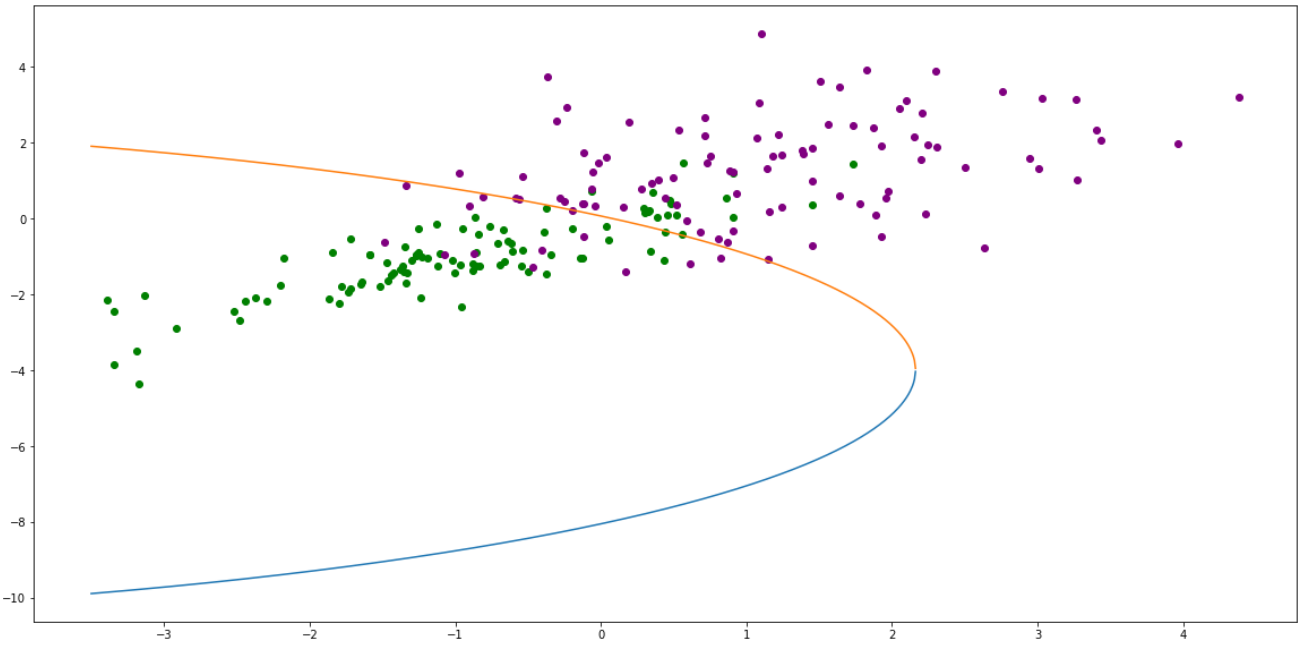
In [744]:

```
A_NB=d_NB-s_NB
def B_NB(x):
    return -2*mu11_NB*d_NB+2*s_NB*mu01_NB+x*b_NB+x*c_NB-mu10_NB*b_NB-mu10_NB*c_N
B-x*q_NB-x*r_NB+mu00_NB*q_NB+mu00_NB*r_NB
def D_NB(x):
    return -((np.log(np.linalg.det(var_cov_NB_Class0))-np.log(np.linalg.det(var_
cov_NB_Class1))-a_NB*(x-mu10_NB)**2+p_NB*(x-mu00_NB)**2-d_NB*(mu11_NB**2)+s_NB*(
mu01_NB**2)+x*mu11_NB*b_NB+x*mu11_NB*c_NB-mu10_NB*mu11_NB*b_NB-mu10_NB*mu11_NB*b
_NB-mu10_NB*mu11_NB*c_NB-x*mu01_NB*q_NB-x*mu01_NB*r_NB+mu00_NB*mu01_NB*q_NB+mu00
_NB*mu01_NB*r_NB))
```

In [745]:

```
fig=plt.figure(figsize=(20,10))
plt.plot(x1_class0_train, x2_class0_train, 'o', color='green')
plt.plot(x1_class1_train, x2_class1_train, 'o', color='purple')
x = np.linspace(-4,4,1000000)
plt.plot(x1, (-B_NB(x1)+np.sqrt((B_NB(x1)**2-4*A_NB*D_NB(x1))))/(2*A_NB))
plt.plot(x1, (-B_NB(x1)-np.sqrt((B_NB(x1)**2-4*A_NB*D_NB(x1))))/(2*A_NB))
plt.show()
```

```
/anaconda3/lib/python3.6/site-packages/ipykernel_launcher.py:5: Runt
imeWarning: invalid value encountered in sqrt
  """
/anaconda3/lib/python3.6/site-packages/ipykernel_launcher.py:6: Runt
imeWarning: invalid value encountered in sqrt
```



In [793]:

```
fig=plt.figure(figsize=(20,10))
plt.plot(x1_class0_valid, x2_class0_valid, 'o', color='green', label="0")
plt.plot(x1_class1_valid, x2_class1_valid, 'o', color='purple', label="1")
plt.yticks(fontsize =27)
plt.xticks(fontsize =25)
plt.xlabel('X1', fontsize=30)
plt.ylabel('X2', fontsize=30)
plt.contour(xx1, xx2, probs, [0.5], linewidths=1, color='blue', label="logistic
- X1 X2")
plt.contour(xx1c[:,:,0,0], xx2c[:,:,0,0], probs_c[:,:,0,0], [0.5], linewidths=1,
color='violet', label="logistic - X1 X2 X1^2 X2^2")
plt.plot(x1, -0.75951*x1+0.24049, color='red', label="LDA")
plt.plot(x1, (-B(x1)+np.sqrt((B(x1)**2-4*A*D(x1))))/(2*A),color='black',label="Q
DA")
plt.plot(x1, (-B(x1)-np.sqrt((B(x1)**2-4*A*D(x1))))/(2*A), color='black', label=
"QDA")
plt.plot(x1, (-B_NB(x1)+np.sqrt((B_NB(x1)**2-4*A_NB*D_NB(x1))))/(2*A_NB), color=
'yellow', label="NB")
plt.plot(x1, (-B_NB(x1)-np.sqrt((B_NB(x1)**2-4*A_NB*D_NB(x1))))/(2*A_NB),color='
yellow', label="NB")

plt.show()
```

```
/anaconda3/lib/python3.6/site-packages/matplotlib/contour.py:1004: U
serWarning: The following kwargs were not used by contour: 'color',
'label'
  s)
/anaconda3/lib/python3.6/site-packages/ipykernel_launcher.py:11: Run
timeWarning: invalid value encountered in sqrt
  # This is added back by InteractiveShellApp.init_path()
/anaconda3/lib/python3.6/site-packages/ipykernel_launcher.py:12: Run
timeWarning: invalid value encountered in sqrt
  if sys.path[0] == '':
/anaconda3/lib/python3.6/site-packages/ipykernel_launcher.py:13: Run
timeWarning: invalid value encountered in sqrt
  del sys.path[0]
/anaconda3/lib/python3.6/site-packages/ipykernel_launcher.py:14: Run
timeWarning: invalid value encountered in sqrt
```