Práctica 2

David Gil Bautista

45925324-M

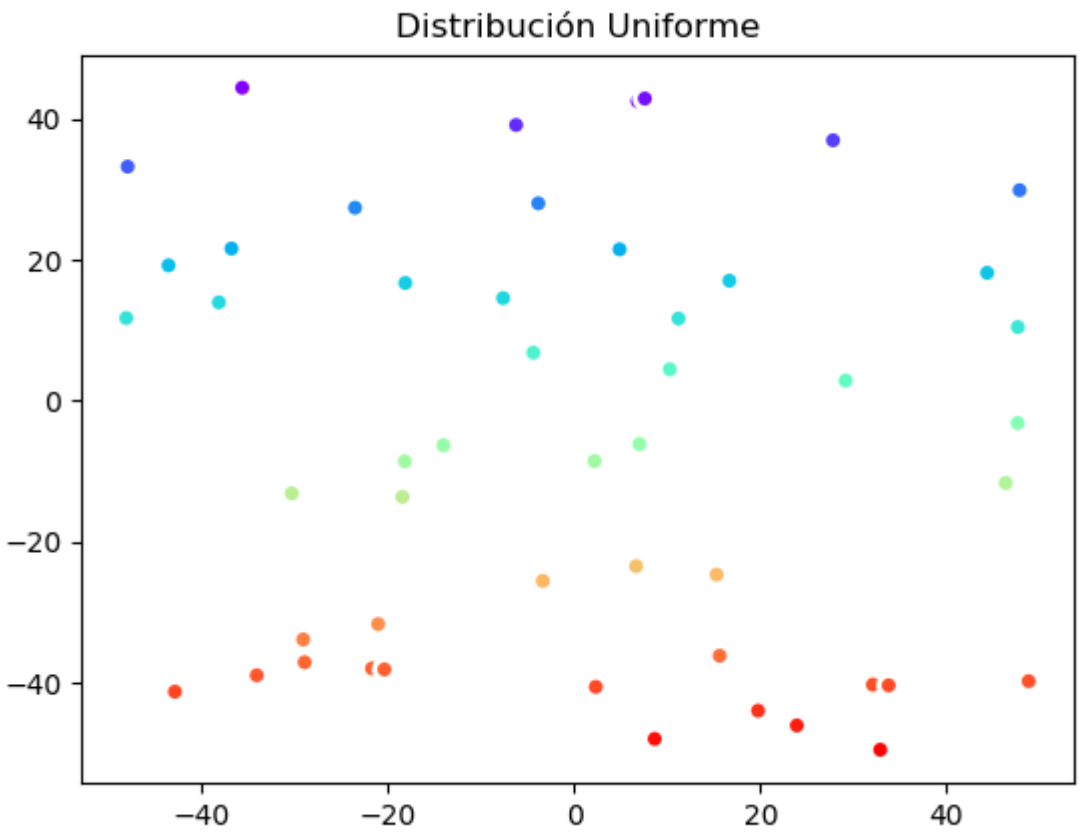
Grupo 1 - Martes 17:30

Ejercicio 1

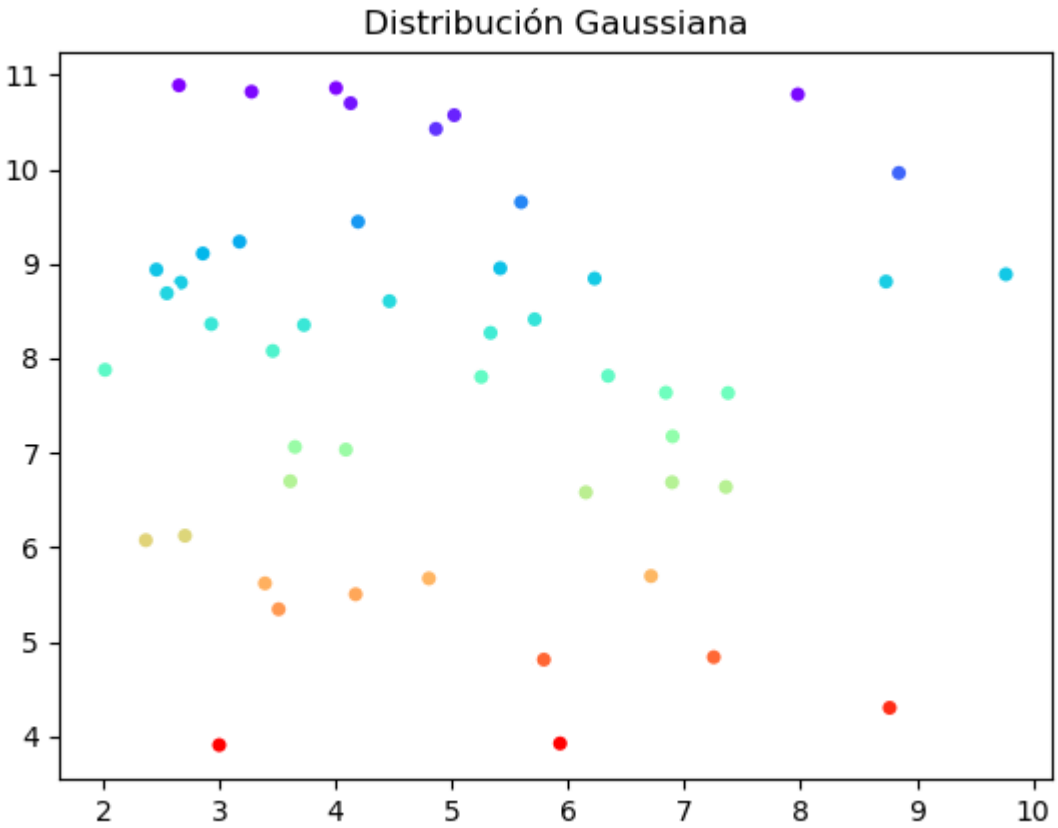
Apartado 1

En este primer ejercicio se nos pide estudiar la complejidad que supone el ruido que tienen las muestras a la hora de escoger una función para clasificar.

Para ello en el primer apartado generamos dos muestras de puntos usando una distribución uniforme y una distribución gaussiana.

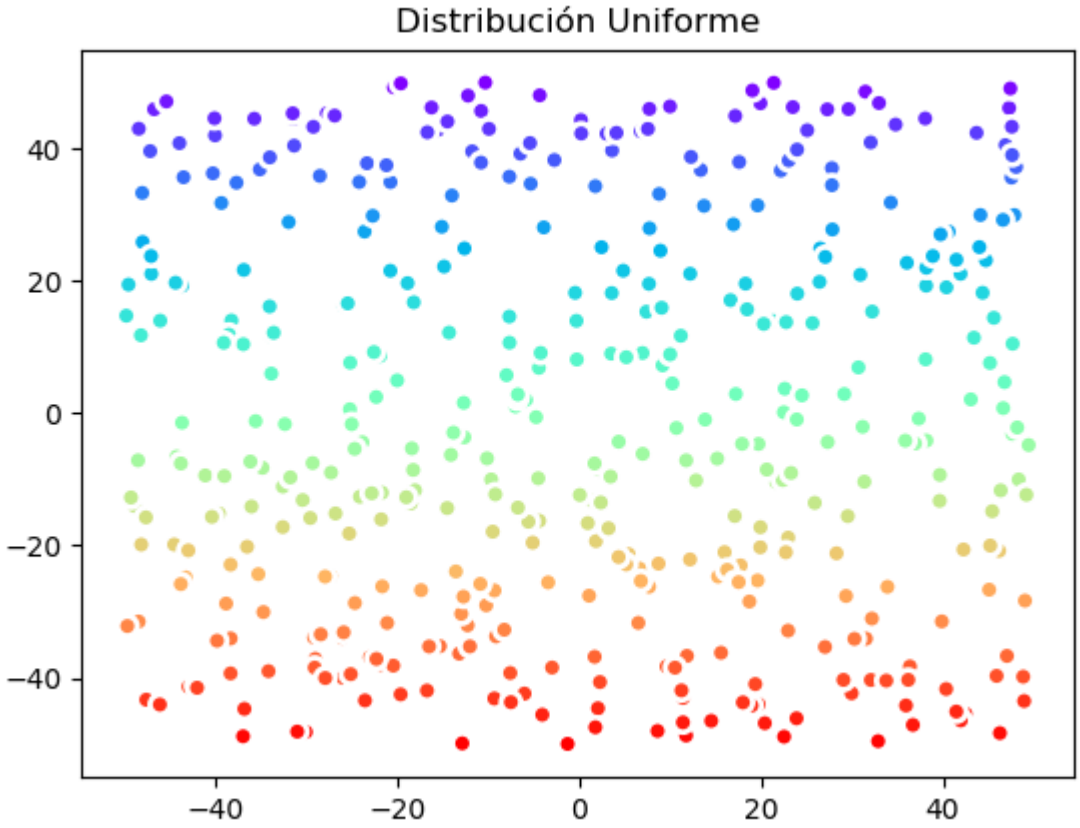


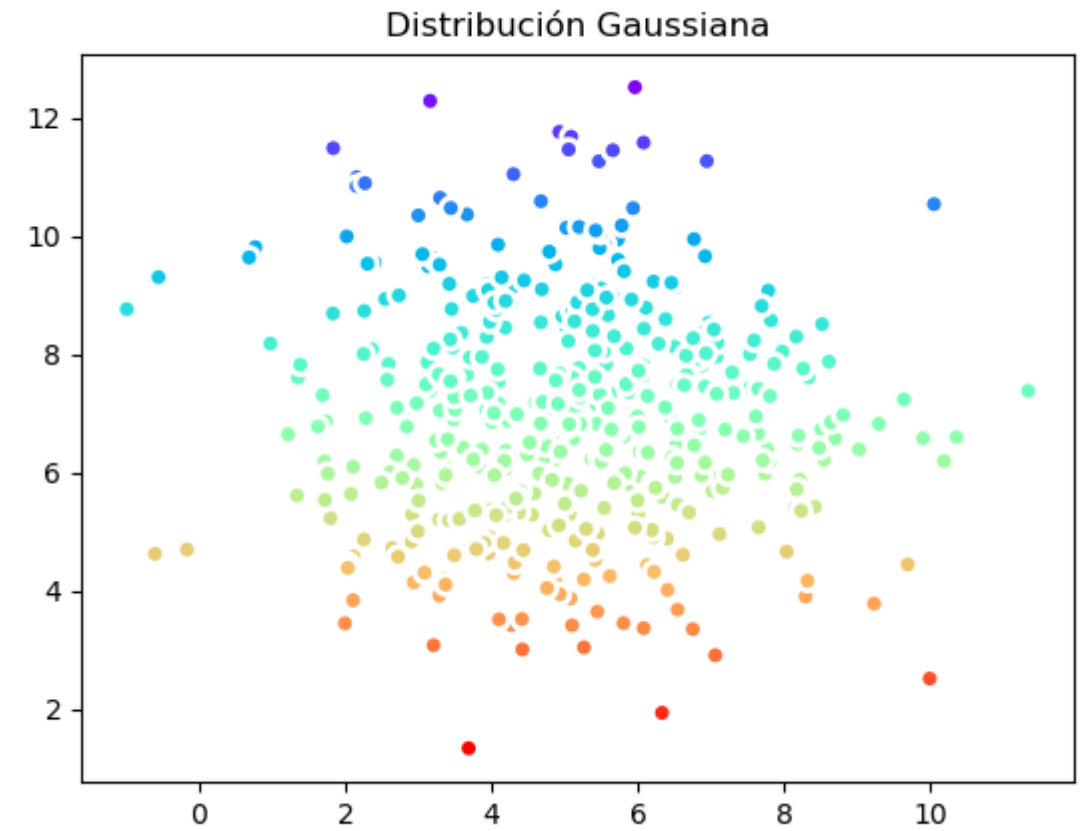
Para la distribución uniforme y un tamaño de 50 podemos apreciar que tenemos los puntos esparcidos por todo el mapa. Sin embargo, con la gaussiana, podemos ver lo siguiente:



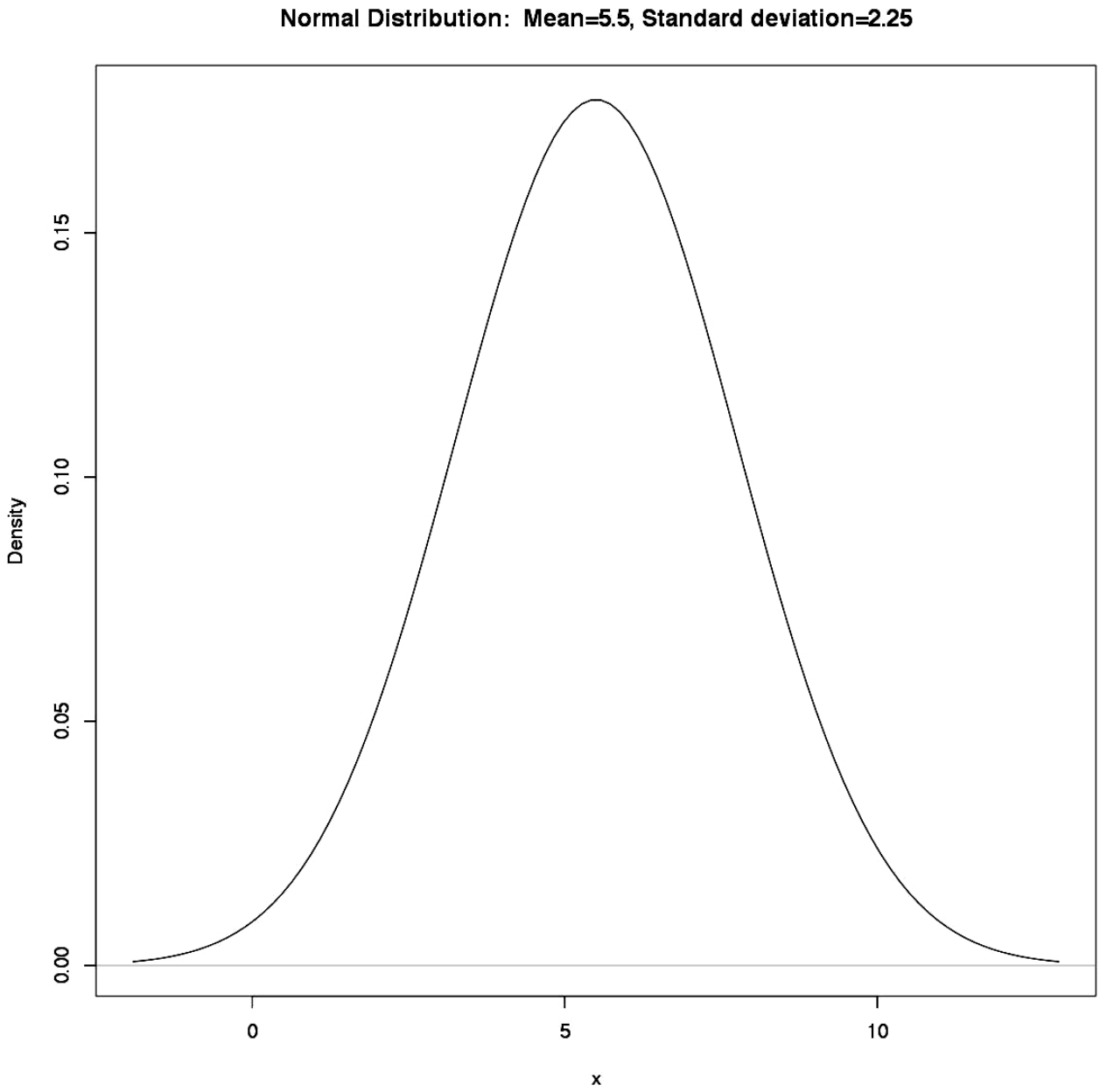
Aquí podemos ver que las muestras tienen tendencia a agruparse más en el centro.

Para comprobar esto de una forma más clara tan sólo debemos aumentar el tamaño de puntos producidos por la distribución.


Distribución Uniforme
40
20
0
−20
−40
−40
−20
0
20
40


Distribución Gaussiana
12
10
8
6
4
2
0
2
4
6
8
10

Ahora si podemos ver claramente que con una distribución gaussiana los puntos tienden a concentrarse en un punto.



Vienda esta función que representa una distribución gaussiana (normal), el máximo de la función sería la moda, que es el valor que más se repite en la gráfica que hemos creado, conforme bajamos por ambos lados de la curva, la probabilidad de que un elemento aparezca disminuye.

Apartado 2

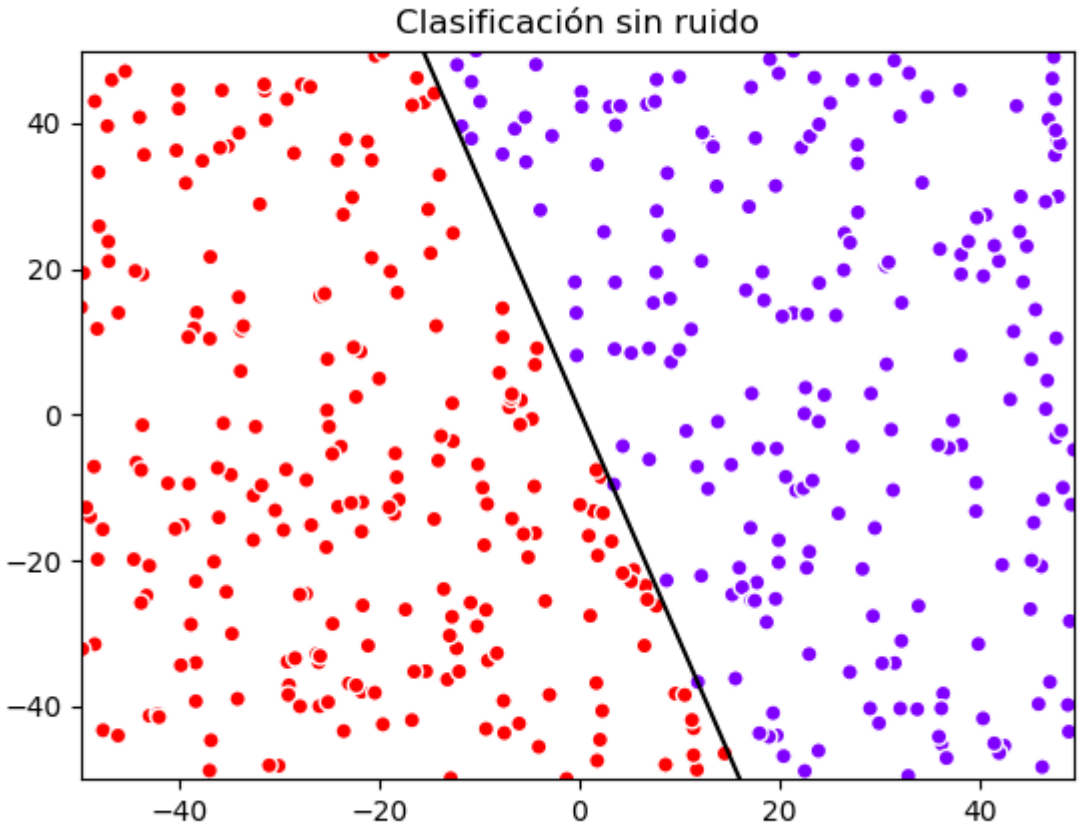
Ahora, a partir de una distribución uniforme crearemos una recta que divida a nuestro conjunto en dos trozos, cada trozo tendrá una etiqueta distinta.

Usando la función *simula_recta* creamos una recta con los siguientes valores:

```
---------------------------- APARTADO 2 ----------------------------
--------------------------------------------------------------------

Recta
a :  -3.15674860025513
b :  0.7385626531371003

Creado conjunto de distribución uniforme con etiquetas medidas mediante la
distancia del punto a la recta.
```

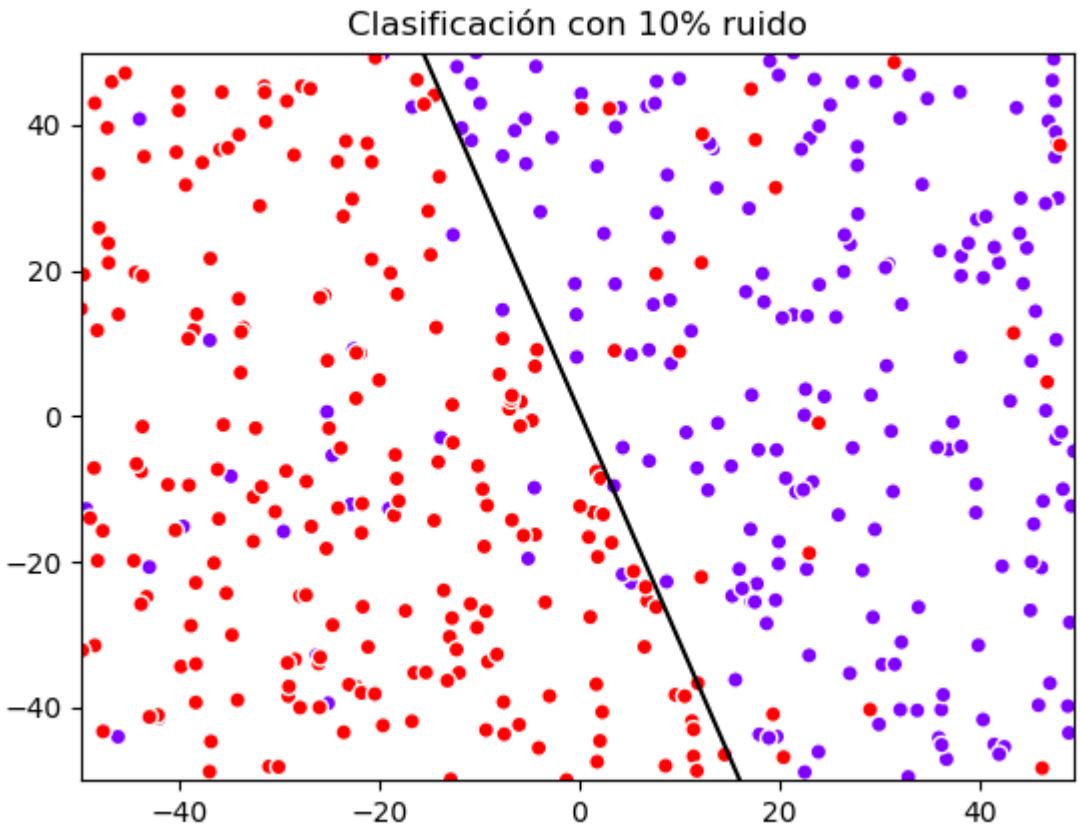
Es decir, la recta es del tipo $y = 0,51x + 0,68$.

Una vez creada la recta y nuestro conjunto de datos dividimos la muestra usando como clasificación la distancia de los puntos a la recta.



Haciendo esto podemos ver que obtenemos una clasificación con un error de 0.

Cambiando ahora un 10% de ambas etiquetas podemos observar lo siguiente:



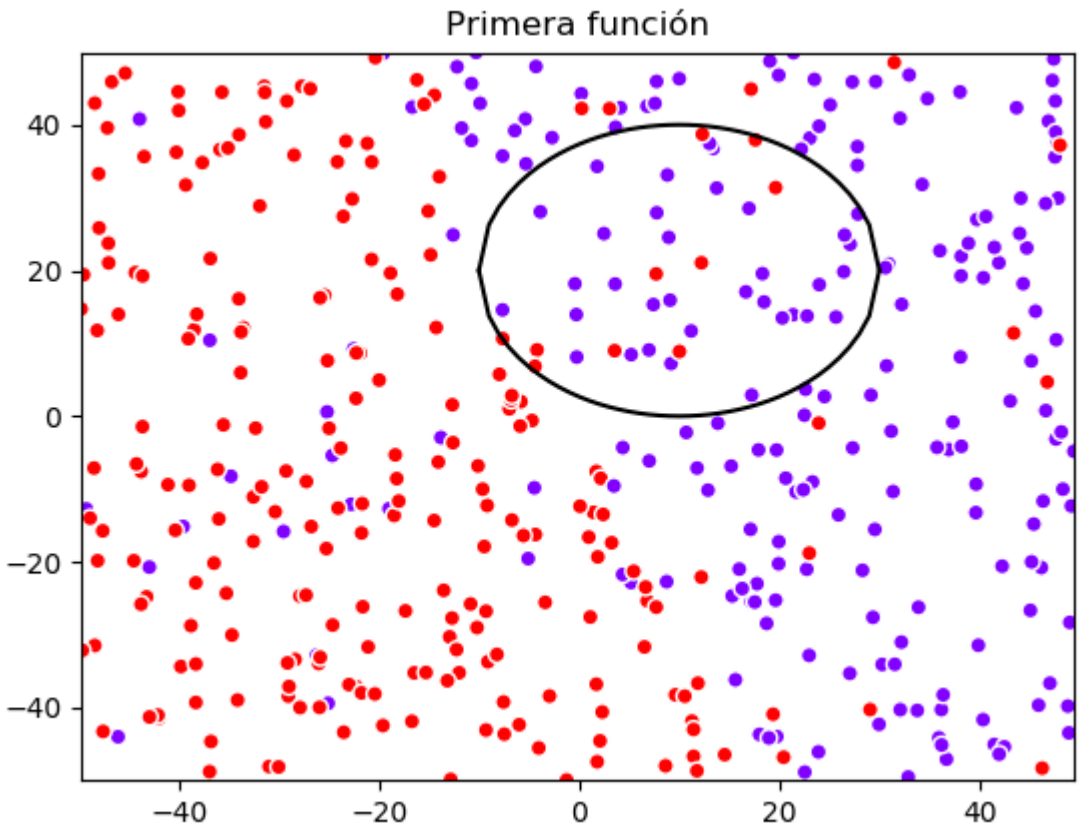
Ahora si que tenemos varios puntos que no están bien clasificados. Al hacer un conjunto con muestras más grandes y clasificar antes de añadir el ruido podemos apreciar que tan solo hay una forma de clasificar bien los elementos, si quisieramos clasificar los elementos ahora tras haber añadido el ruido no habría ninguna función que nos permitiera clasificar correctamente la muestra, lo óptimo sería usar la recta que hemos obtenido al principio y obviar los puntos con ruido.

Apartado 3

Para dar más peso a lo expuesto en el apartado anterior intentaremos clasificar con distintas funciones para comprobar si consiguien dividir el conjunto de una mejor o peor forma.

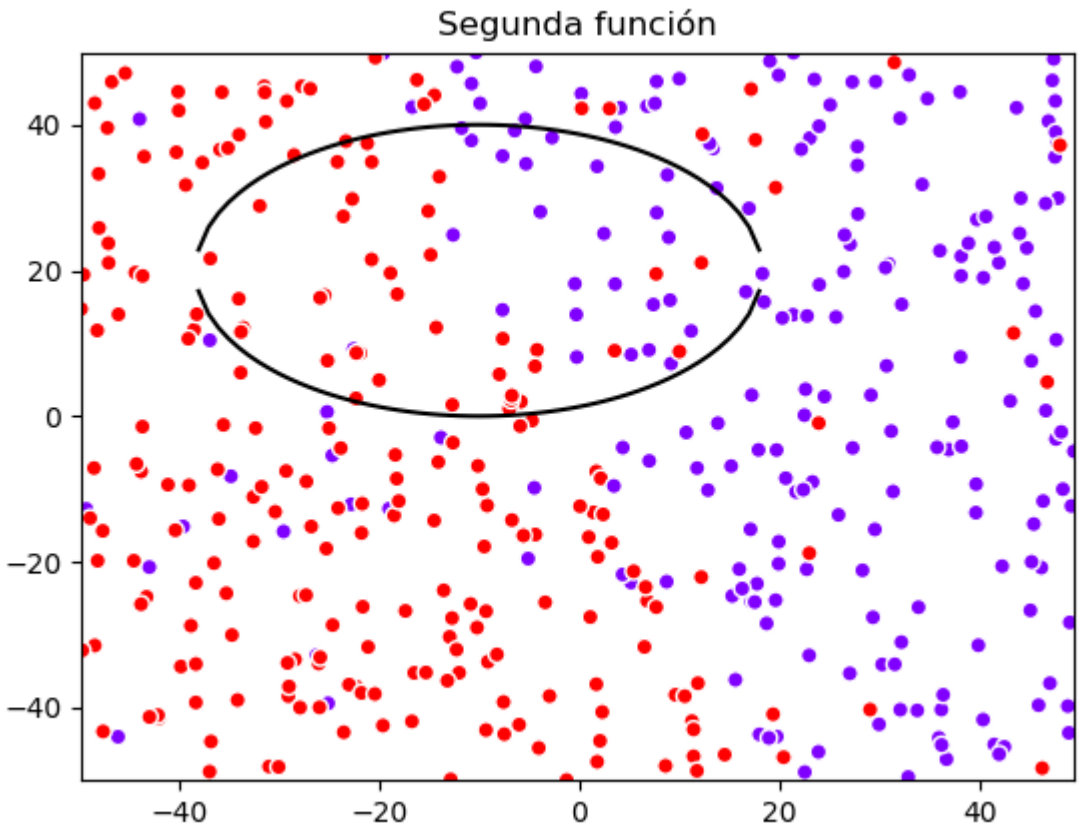
Como se trata de funciones con dos variables hemos despejado la variable y para ajustarlas a un plano 2D.

a) $f(x, y) = (x - 10)^2 + (y - 20)^2 - 400$



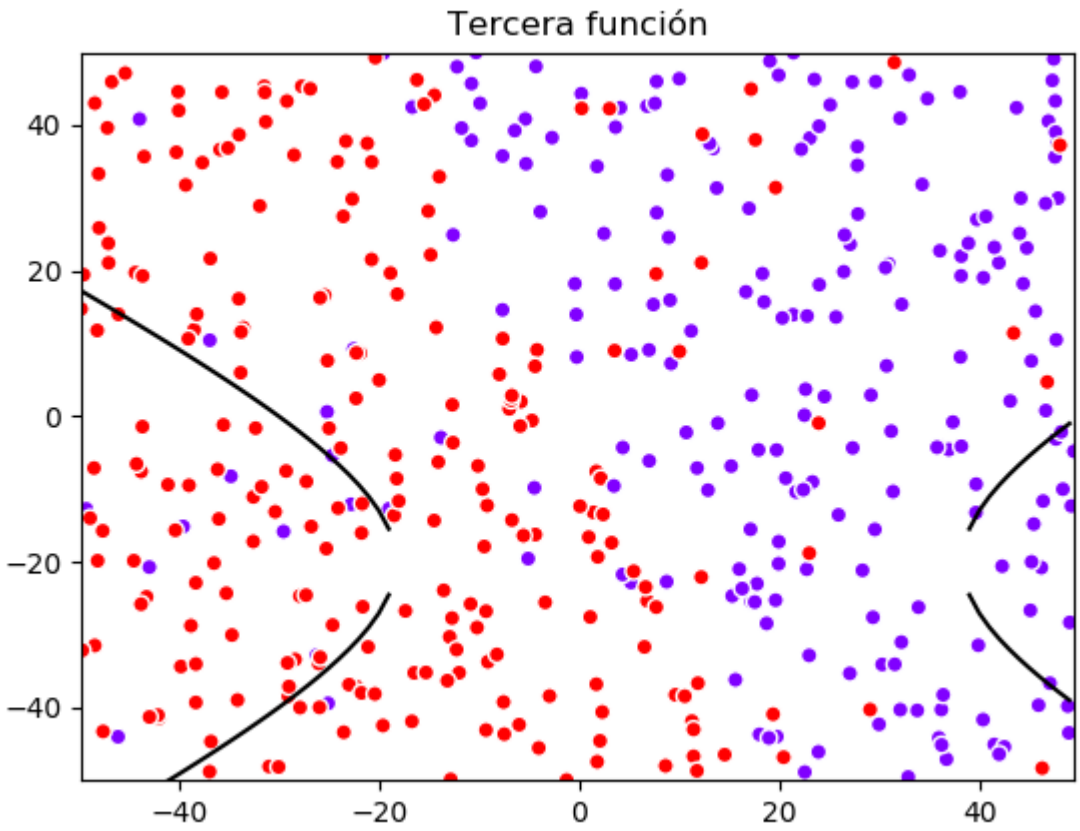
Con esta función podemos ver que captamos una parte del conjunto que tiene las muestras azules, despreciando la mayoría de las muestras rojas. Aunque parezca que divide bien el conjunto se puede ver que está desechando muchas muestras que debería haber cogido también.

b) $f(x, y) = 0,5(x + 10)^2 + (y - 20)^2 - 400$



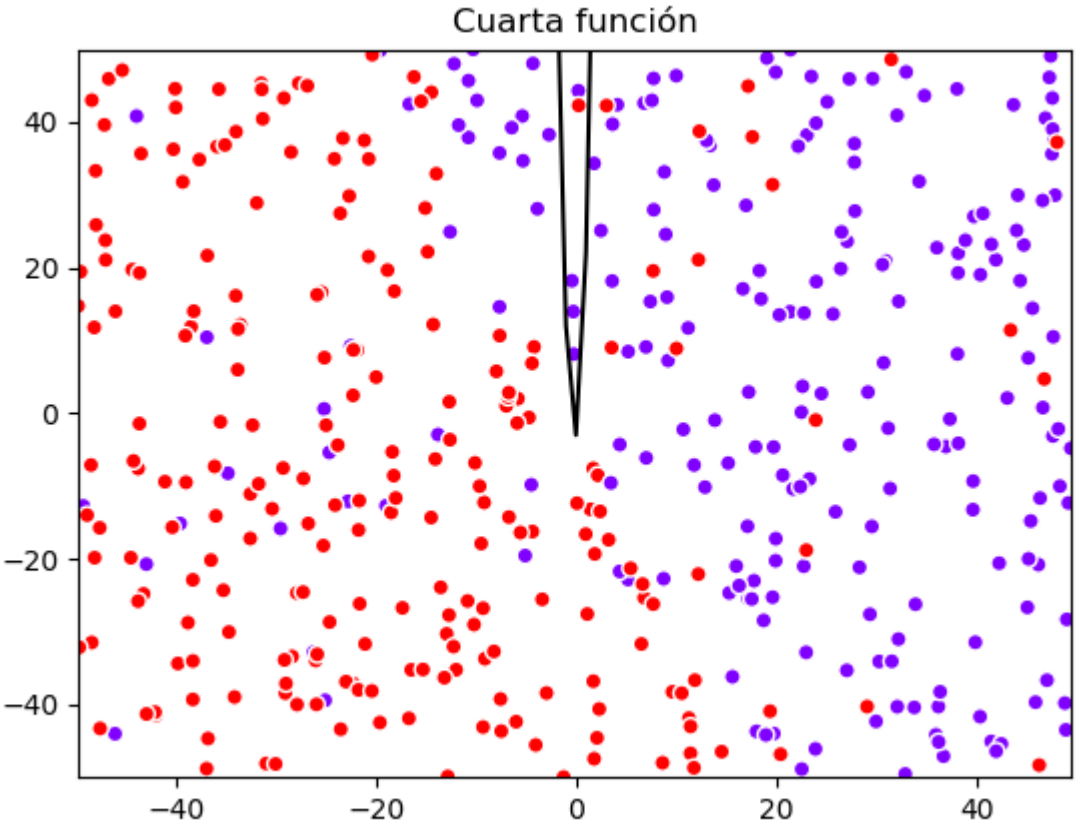
En este ejemplo (aunque no se pueda apreciar debido a un error técnico) podemos ver que al expandirse el círculo agrupa a un mayor número de muestras azules, sin embargo, al igual que en el anterior, desecha varias muestras que debería recoger.

c) $f(x, y) = 0,5(x - 10)^2 - (y + 20)^2 - 400$

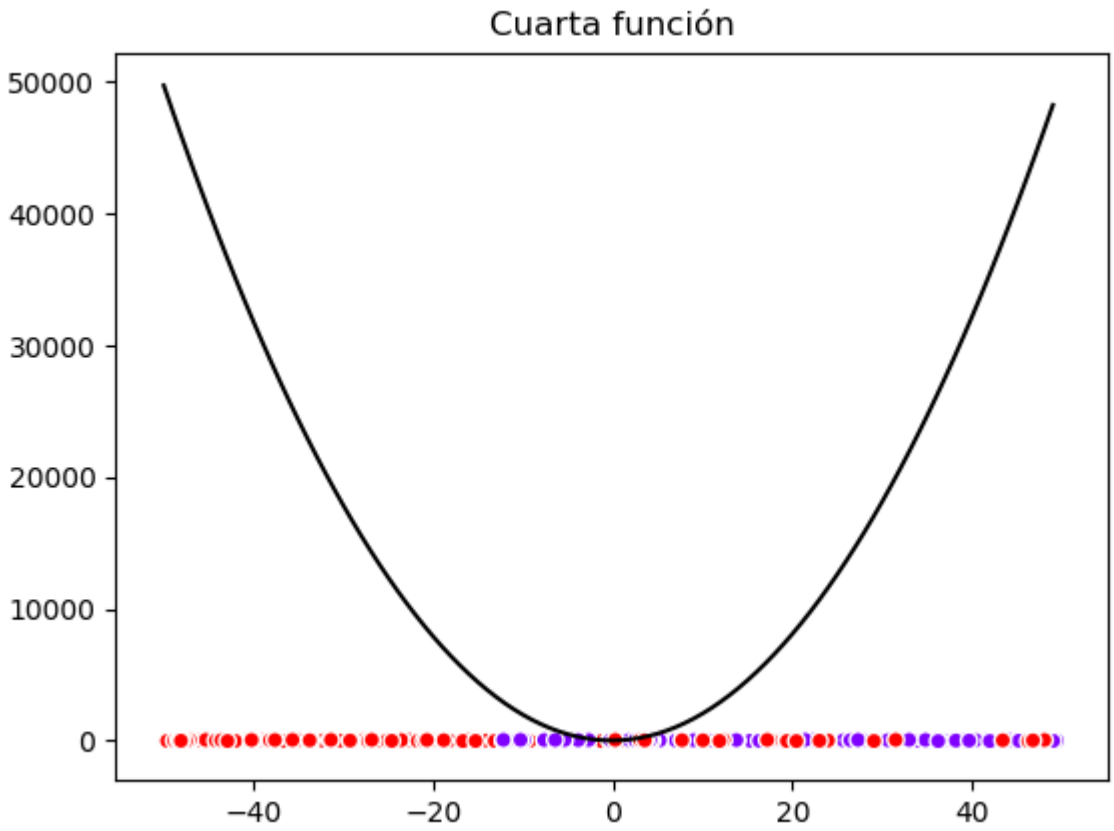


Con esta hiperbola se puede apreciar que ahora recoge una no tan grande cantidad de muestras rojas, aunque sigue desechando la mayoría.

d) $f(x, y) = y - 20x^2 - 5x + 3$



Esta función es una parábola y debido al minúsculo tamaño que ocupa el conjunto de muestras tan sólo podemos ver el mínimo de la parábola, el cual agrupa la mayoría de puntos de la muestra azul, pero no llega ni al 10% del total de puntos. Si aumentamos el tamaño de la gráfica podemos ver la forma real que tiene nuestra función.



Como conclusión he observado que para determinados problemas, en este caso una distribución uniforme, una función no lineal no tiene porqué dividir mejor al conjunto que una lineal (esto es así porque hemos clasificado usando una lineal). Es decir, aunque a veces aumentemos la complejidad de la clasificación, puede que la respuesta sea mucho más sencilla.

Ejercicio 2

Apartado 1

Para la primera parte de este ejercicio se nos pide diseñar el algoritmo del perceptron para la clasificación binaria. Lo hemos hecho de la siguiente forma:

```
def ajusta_PLA(datos, label, max_iter, vini):

    w = vini.copy()

    iteraciones = max_iter
```

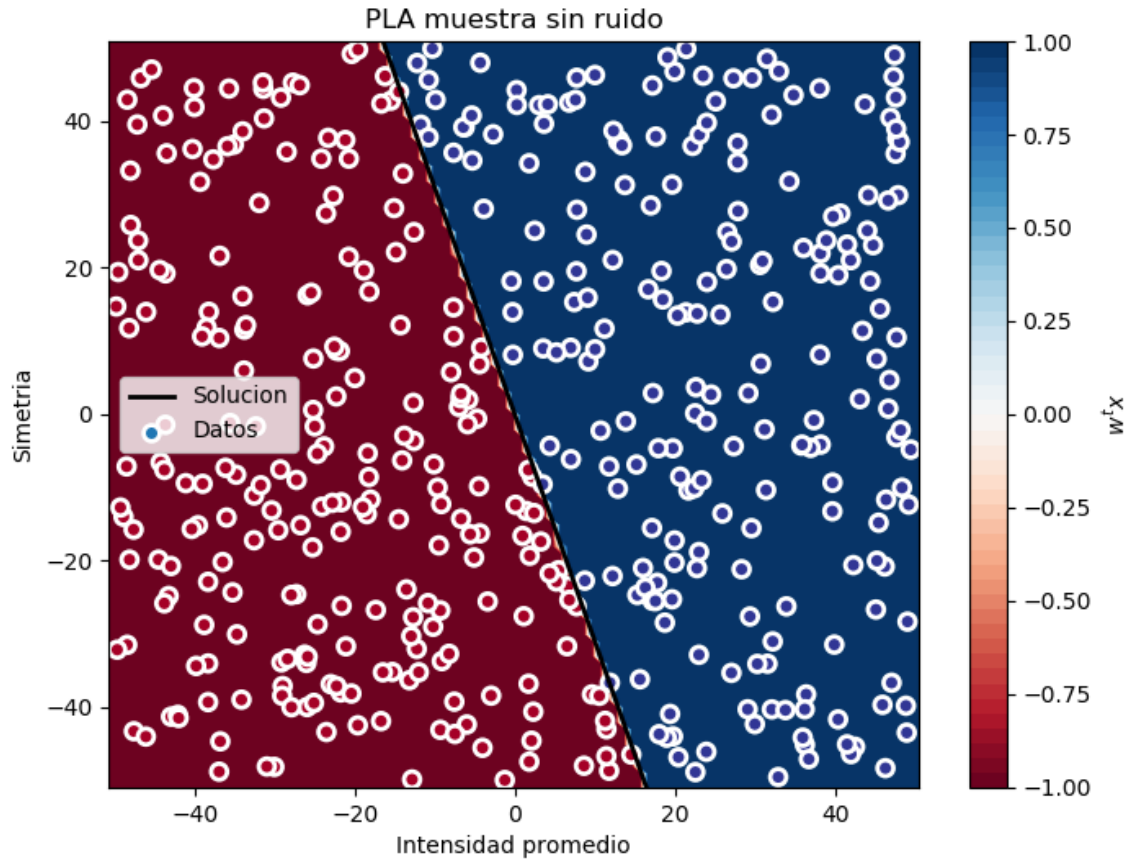
```
    for i in range(max_iter):
        w_copy = w.copy()


        for X,y in zip(datos,label):

            if(np.sign(np.dot(X,w)) != y):
                w += X*y

        if np.all(w_copy == w):
            iteraciones = i+1
            break

    print('El número de iteraciones ha sido {}'.format(iteraciones))
    return w, iteraciones
```

En este algoritmo estamos aprendiendo hasta que el vector de pesos no cambia o hasta llegar al número máximo de iteraciones.

Aplicando este algoritmo a nuestra muestra del apartado 2.a del ejercicio 1 y usando un vector inicial de ceros hemos obtenido lo siguiente:

```
--------------------------- APARTADO 1A -----------------------------

Recta
a :  -3.15674860025513
b :  0.7385626531371003
Con vector inicial =  [0. 0. 0.]

El número de iteraciones ha sido 2
Modelo obtenido =  [85.43579898 27.56886736  0.        ]

De un conjunto de datos de 444 elementos ha fallado en predecir 0
ErrorClasificacion = 0.0
```

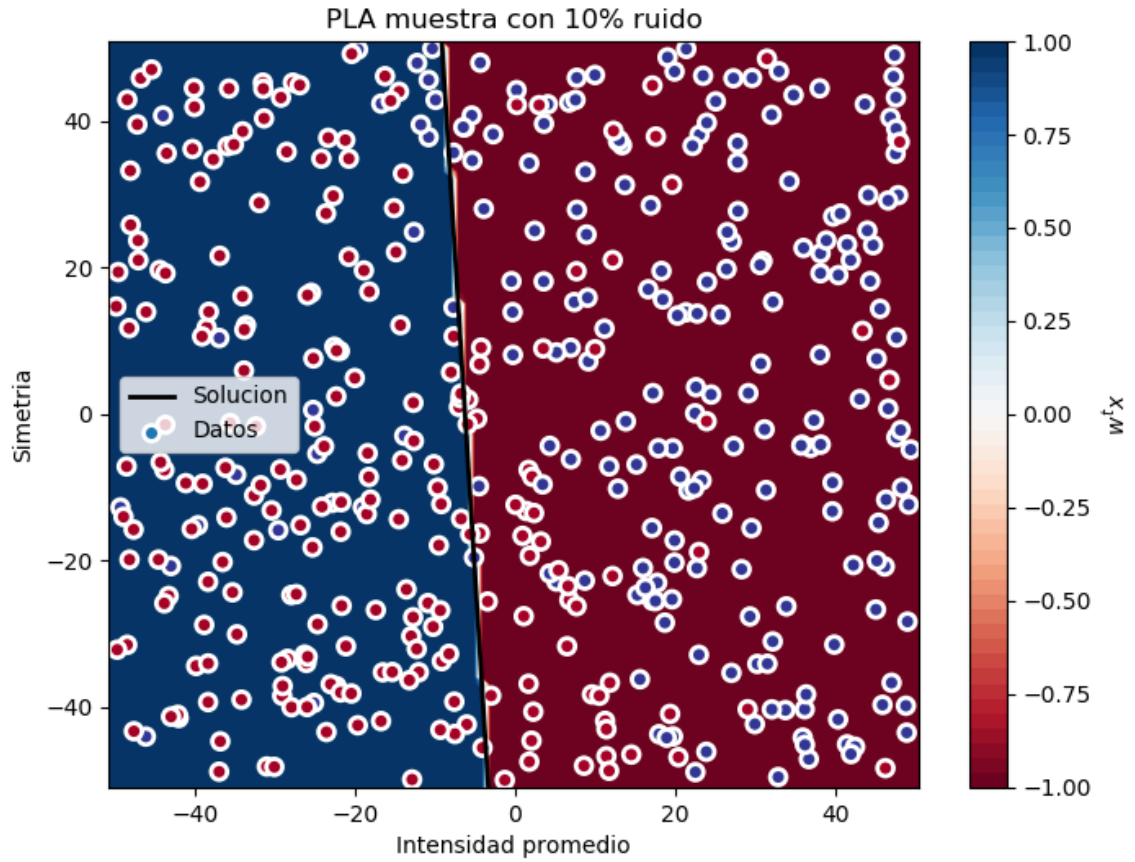
Y como podemos ver, para un conjunto de 444 elementos los clasificamos todos correctamente, esto lo podemos apreciar mejor con una gráfica.



En el segundo apartado se nos pide ejecutar el mismo algoritmo pero utilizando el conjunto de muestras al que le hemos añadido un 10% de ruido. En este apartado obtenemos la siguiente solución:

```
--------------------------- APARTADO 1C -----------------------------

Con vector inicial =  [0. 0. 0.]

El número de iteraciones ha sido 100
Modelo obtenido =  [ -33.70987459   -1.92321104 -209.        ]

De un conjunto de datos de 444 elementos ha fallado en predecir 361
ErrorClasificacion = 0.8130630630630631
```

Aquí podemos ver que en este caso ha conseguido un error del 81%, lo más razonable sería pensar que si al conjunto le hemos añadido un 10% de ruido, éste porcentaje sería el que obtuviéramos de error. Esto no es así ya que solo tendríamos dicho error en caso de que nuestro algoritmo hubiera aproximado la misma solución que la aportada con el conjunto sin ruido. Lo podemos ver mejor en la siguiente gráfica.



Se puede ver que no es la misma recta, y que en la parte roja, la mayoría de los puntos son azules, lo que debería determinar que esa parte fuera la azul.

Para consolidar que este algoritmo funciona bien cuando tenemos datos sin ruido hemos hecho una prueba en la que a partir de la muestra original (sin ruido) y vectores iniciales aleatorios hemos conseguido los siguientes resultados:

```
Con vector inicial 9 = [0.010374153885699955, 0.5018745921487388,
0.4957732931341461]
El número de iteraciones ha sido 2
Modelo obtenido =  [68.71628676 22.00538102  2.49577329]

De un conjunto de datos de 444 elementos ha fallado en predecir 0
ErrorClasificacion = 0.0
------------------------------------------------------------------
Para 10 vectores aleatorios obtenemos una media de iteraciones de 3.3
```

Para 10 vectores aleatorios obtenemos una media de 3.3 iteraciones para encontrar una solución que nos da un error de un 0%. Ya que lo que modelamos es una recta que divide un conjunto, con pocas iteraciones conseguimos unos valores que se ajustan a la recta (dos puntos) que usamos previamente para clasificar el conjunto.

Apartado 2

Para este apartado se nos pide implementar el algoritmo de regresión logística utilizando el gradiente descendente estocástico. Para este algoritmo se nos dan unas indicaciones a tener en cuenta, las cuales son: inicializar el modelo a 0, permutar el conjunto de datos, y que la diferencia entre los modelos sea menor a 0,01 antes de ofrecer una solución.

Dado que vamos a recorrer todo el conjunto de datos, desordenarlo para calcular el gradiente es una estúpidez, lo único que conseguimos es variar nuestro modelo de forma distinta.

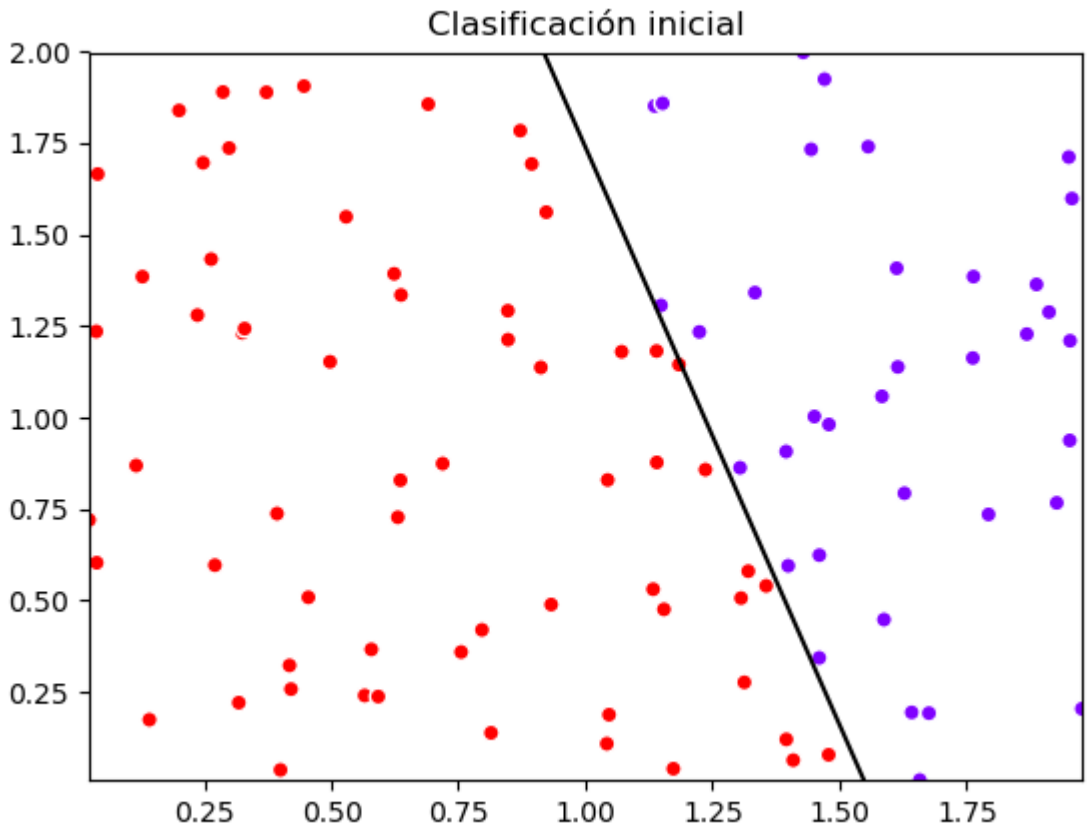
El recorrer los datos usando minibatches también es absurdo puesto vamos a recorrer todo el vector antes de comprobar si hemos hallado ya la solución.

Teniendo en cuenta lo dicho, he dejado el algoritmo tal y cómo se nos pidió implementar en la sesión anterior pero añadiendo las condiciones que se nos pedía.

La única modificación relevante que se ha realizado ha sido el cambio del gradiente para el cual se ha usado la función sigmoide.

He probado a hacer el gradiente sin hacer la media del conjunto y la tasa de aciertos aumenta en un 4-5%. Es posible que para determinados conjuntos una función sea mejor que otra y por esto en el caso de no usar la media mejora la predicción. Aunque al hacer esto nuestro modelo clasifique mejor he optado por dejarlo tal y cómo se pide.

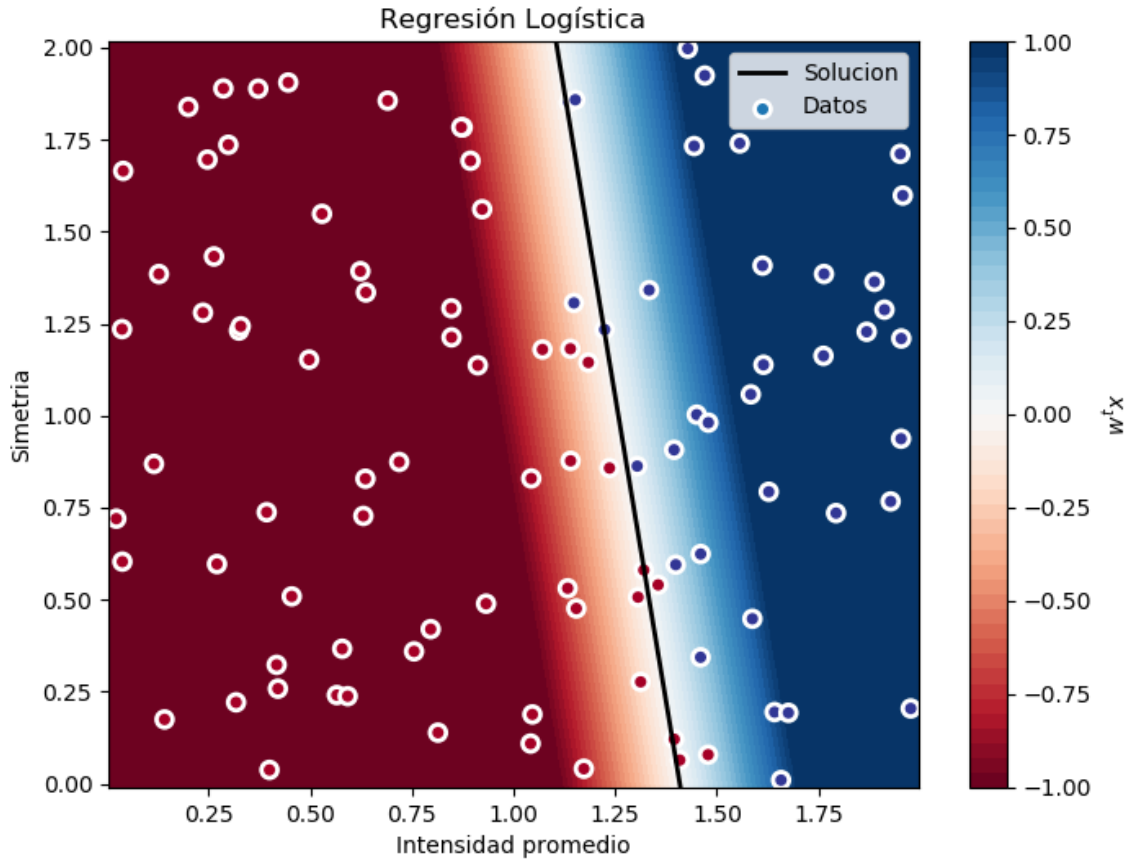
Hemos partido de una muestra uniforme de 100 puntos y una recta tomados en el intervalo [0,2] y al igual que en el ejercicio anterior los hemos etiquetado, aunque en esta ocasión no hemos añadido ruido.




```
---------------------------- APARTADO 2A -----------------------------


Recta
a = -3.15674860025513
b = 4.89531125339223

Modelo inicial:  [0. 0. 0.]

Calculando regresión logística para 1000 iteraciones.
Please, do not turn off your computer

Modelo terminado con 311 iteraciones
Modelo tras entrenar:  [ 3.36768042  0.50813659 -4.74461232]
Porcentaje de éxito = 95.0%
```

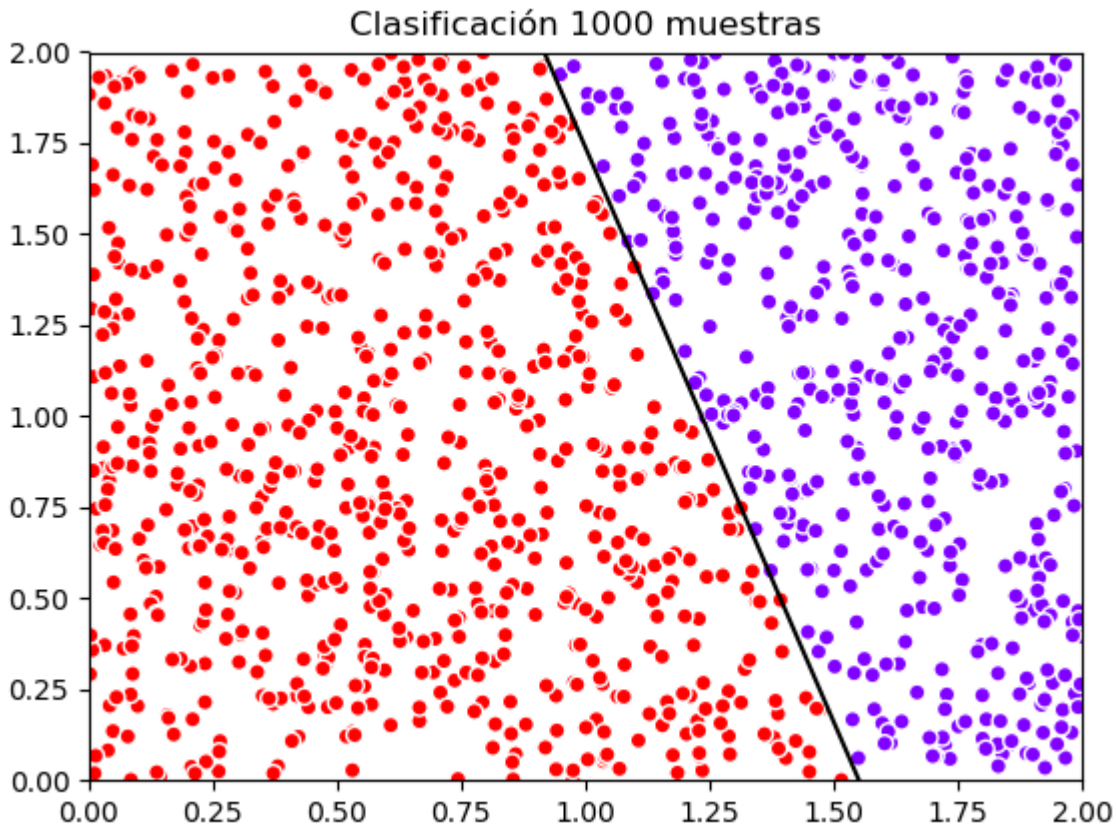
Al usar nuestro modelo de regresión podemos comprobar que en pocas iteraciones llegamos a conseguir un modelo que nos garantiza un 95% (99% en caso de cambiar el gradiente) de aciertos, dado que es un modelo de regresión y hemos establecido una parada no exacta es posible que no consigamos acertar el conjunto completo, si en nuestra condición de parada fuéramos más restrictivos podríamos llegar al 100% de aciertos

aunque tardaría un poco más, sin embargo, con dicha restricción y número de iteraciones, obtenemos una muy buena solución, esto lo podemos ver en la siguiente gráfica:



Como se puede ver, el ajuste no nos proporciona la misma recta que hemos usado para clasificar los datos al principio, aún así, sabemos que esta es una solución correcta.

Para hacer un mejor estudio de los errores hemos generado un conjunto con más muestras y las hemos clasificado usando la misma recta, obtenemos la siguiente muestra de datos:

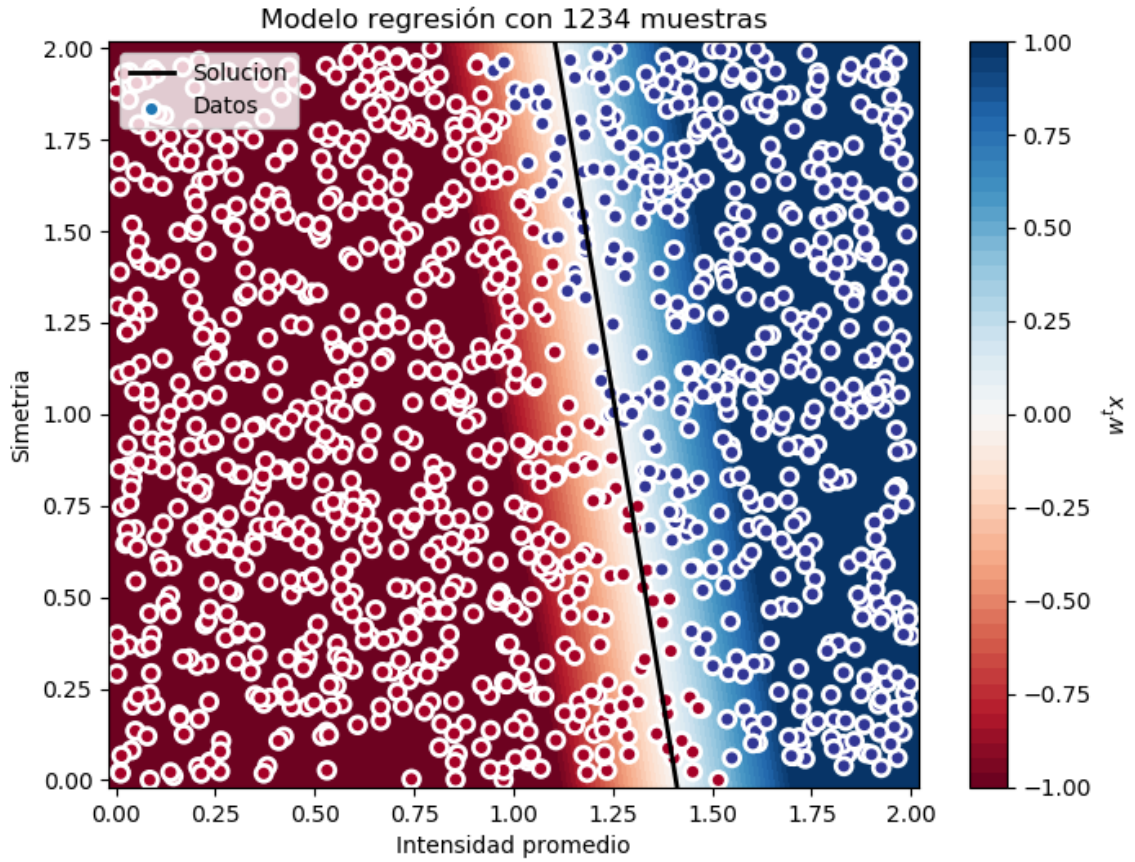


Ahora se puede observar que hay muchos más puntos pegados a la recta, lo que puede provocar que al no tener un buen ajuste el error crezca más.

Usando nuestro modelo ajustado previamente obtenemos el siguiente porcentaje de aciertos:

```
--------------------------- APARTADO 2B -----------------------------

Generadas 1234 muestras de una distribución uniforme.

Aplicamos nuestro modelo calculado anteriormente a la nueva muestra.

Modelo:  [ 3.36768042  0.50813659 -4.74461232]
Porcentaje de éxito = 95.54294975688818%
```

Al tener más datos en este conjunto obtenemos un porcentaje mayor, en el otro conjunto al solo tener 100 datos obtenemos 95%, en este conjunto aumentamos este porcentaje un poco más. Al haber aumentado el conjunto de datos usando una distribución uniforme se garantiza que nuestro modelo de regresión ha ofrecido una solución capaz de dividir bastante bien al conjunto de datos, lo vamos a ver en la siguiente gráfica:



Se puede ver que aquellos datos muy próximos a la frontera de clasificación se han clasificado mal debido a que el modelo no ha ofrecido la misma recta con la que se clasificaron previamente. De todas formas, al ser una muestra uniforme y tener tantos datos, que unos pocos no se clasifiquen correctamente no influyen tanto puesto que estos tienen menos peso.

La conclusión que se saca de esta sesión es que la correcta clasificación de los datos nos garantiza obtener una muy buena solución a la hora de usar un algoritmo de aprendizaje automático para obtener un modelo que nos permita clasificar los datos, y que una pequeña modificación en alguno de estos algoritmos te permite encontrar una solución más o menos óptima.