Práctica 2
Programación

Aprendizaje Automático - GII
Grupo 2
Curso 19-20

Universidad de Granada
Cumbreras Torrente, Paula
49087324-B

Índice de contenidos

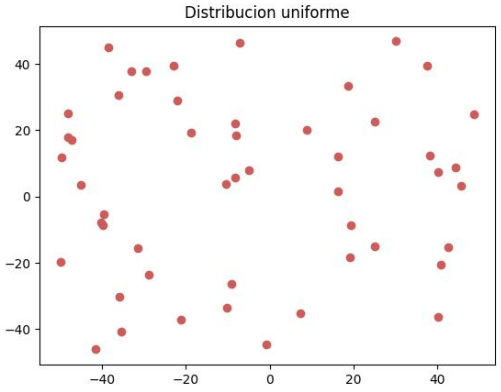
Ejercicio sobre la complejidad de H y el ruido

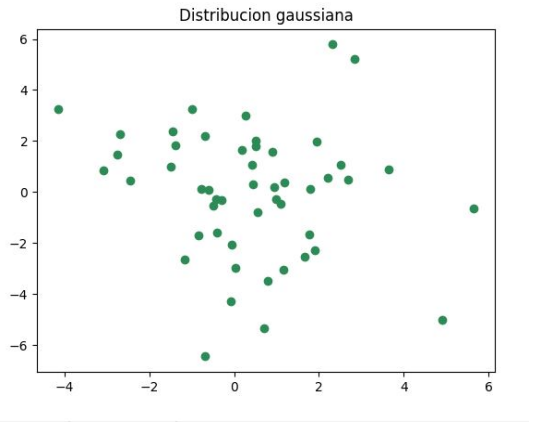
Apartado 1

Trabajaremos con 3 funciones ya implementadas:

- Función simula_unif(N, dim, rango) genera una lista de N vectores de dimensión dim con números aleatorios uniformes dentro del intervalo rango.
- Función simula_gaus(N, dim, sigma) genera una lista de N vectores de dimensión dim con números aleatorios extraídos de una distribución Gaussiana de media 0 y varianza dada por la posición del vector sigma.
- Función simula_recta(intervalo) genera de forma aleatoria los parámetros, v=(a, b) de una recta, y=ax+b, que corta al cuadrado intervalo X intervalo.

Para este apartado se pide generar dos nubes de puntos, una con distribución uniforme y otra con distribución Gaussiana, y representarlas en una gráfica.



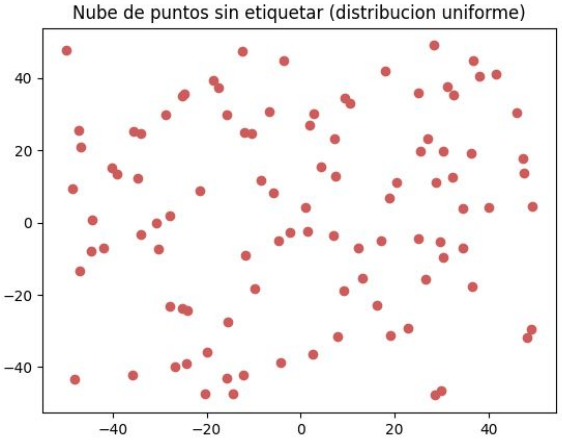


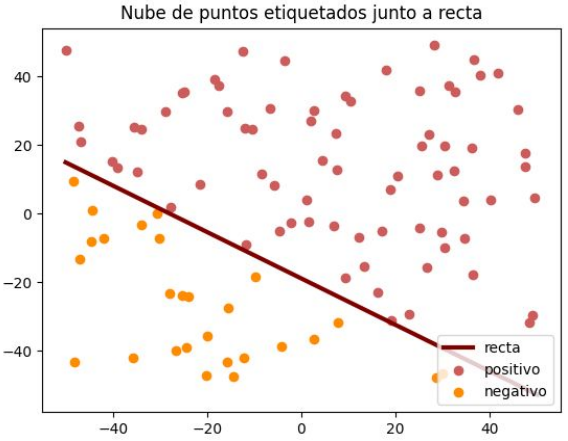
Para la distribución uniforme se han generado 50 vectores de dimensión 2 con números aleatorios uniformes comprendidos por el intervalo [-50, 50]. Para la distribución Gaussiana, se han generado también 50 vectores de 2 dimensiones y sigma [5, 7].

Apartado 2

Para este apartado trabajaremos sobre una nube de 100 puntos distribuidos uniformemente, a los que les asignaremos una etiqueta en función de la distancia de cada punto a una recta que simularemos con simula_recta en el intervalo [-50, 50]. Para ello, implementamos la función f(x, y, a, b)

```
calcular distancia entre punto y recta
si distancia es mayor o igual que 0 devolver 1
si distancia es menor que 0 devolver -1
```

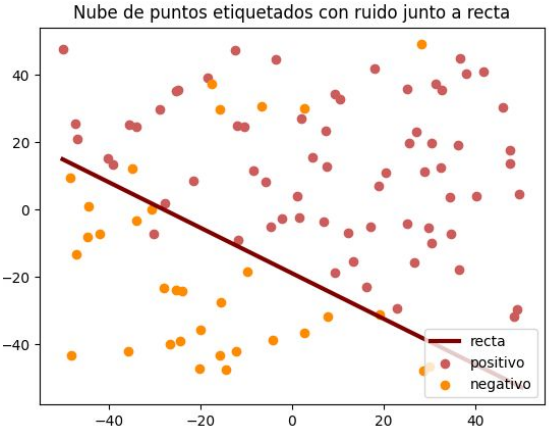




A continuación le añadiremos ruido a la nube de puntos cambiando la etiqueta de un 10% de los puntos etiquetados positivamente y de un 10% de los etiquetados negativamente. Hacemos uso de la función aniadir_ruido(etiquetas)

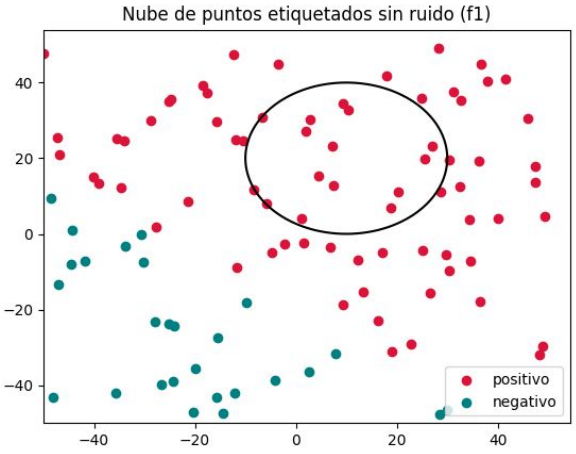
```
separar las etiquetas en positivas y negativas
seleccionar aleatoriamente un 10% de positivas
cambiar su etiqueta a -1
seleccionar aleatoriamente un 10% de negativas
cambiar su etiqueta a 1
devolver etiquetas
```

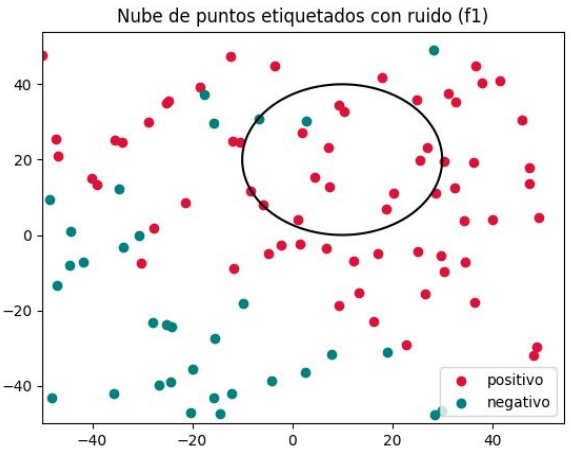
En la gráfica de la derecha podemos ver cómo queda la nube de puntos al añadirle ruido a las etiquetas. Como se puede observar, al añadir ruido los puntos quedan mal clasificados, habiendo puntos positivos por debajo de la recta divisoria y puntos negativos por encima de la misma. Ya que el objetivo de este ejercicio es comprender cómo afecta el ruido a la hora de escoger la clase de funciones, trabajaremos sobre la misma nube de puntos pero modificando la función que hemos usado para etiquetar los puntos.



- La primera función utiliza como frontera de la clasificación de puntos una circunferencia.

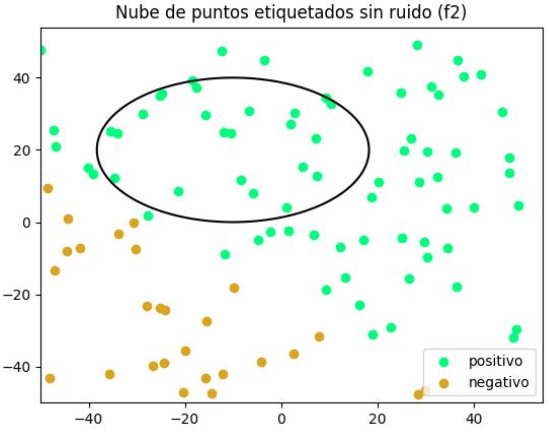
$$f1(x, y)= (x-10)^2+(y-20)^2-400$$

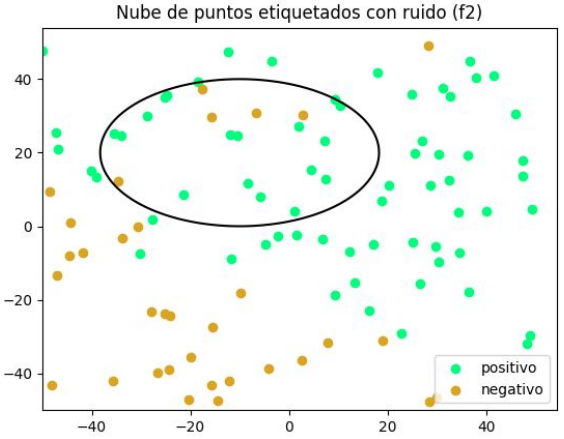




- La segunda función utiliza como frontera de la clasificación de puntos una elipse.

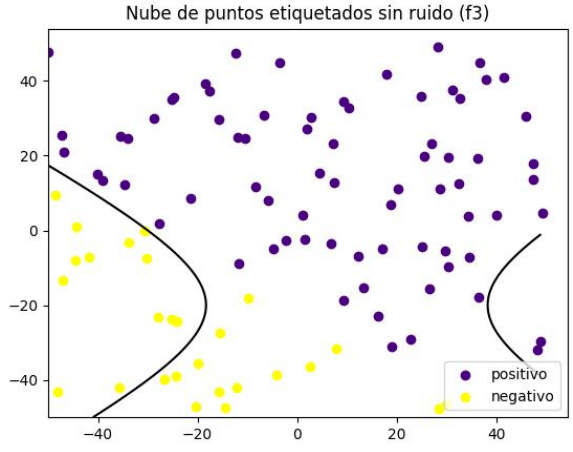
$$f2(x, y) = 0{,}5(x+10)^2+(y-20)^2-400$$

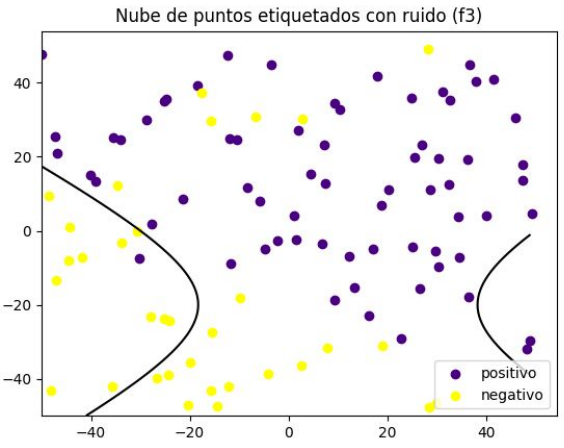




- La tercera función utiliza como frontera de la clasificación de puntos una hipérbola.

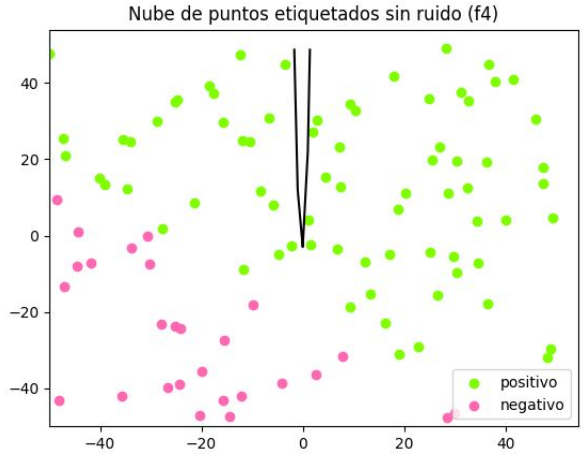
$$f3(x, y) = 0{,}5(x-10)2 -(y+20)2 -400$$

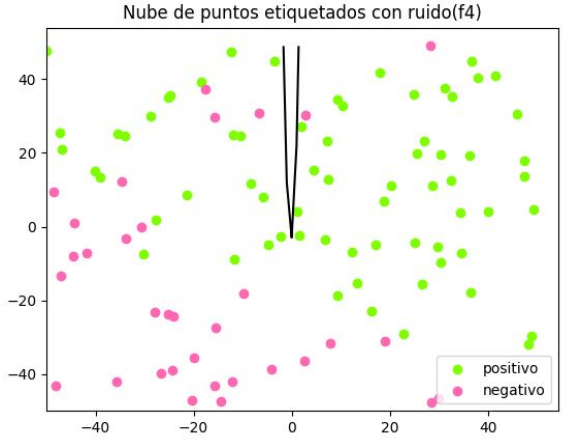




- La cuarta función utiliza como frontera de la clasificación de puntos una parábola.

f4(x, y)= y−20x2−5x+3





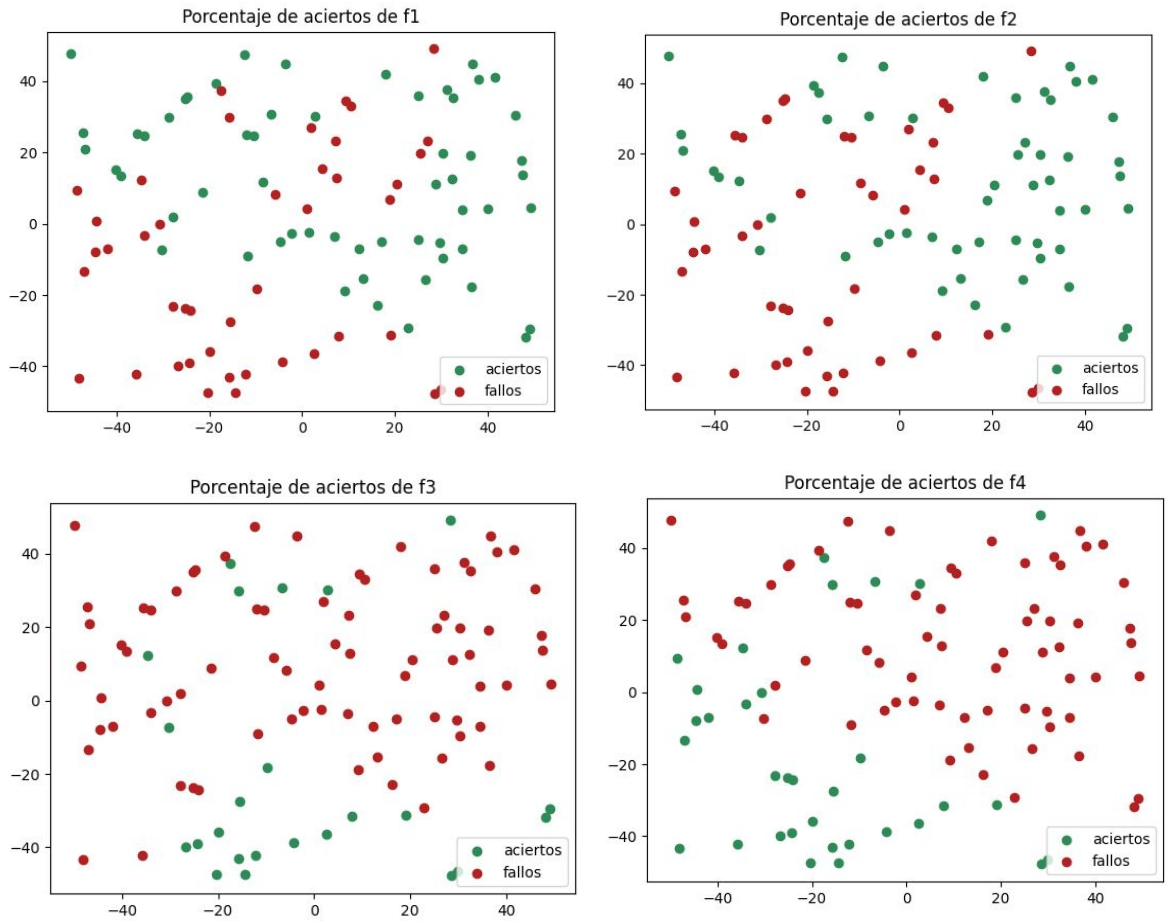
---------- Sin ruido ----------

Porcentaje de aciertos de la función 1: 59 %
Porcentaje de aciertos de la función 2: 51 %
Porcentaje de aciertos de la función 3: 16 %
Porcentaje de aciertos de la función 4: 27 %

---------- Con ruido ----------

Porcentaje de aciertos de la función 1: 57 %
Porcentaje de aciertos de la función 2: 55 %
Porcentaje de aciertos de la función 3: 24 %
Porcentaje de aciertos de la función 4: 33 %

La muestra está clasificada de manera lineal, por lo que ninguna función, por muy compleja que sea, delimitará mejor la nube de puntos que la función lineal. La primera función, como podemos ver en los porcentajes de aciertos mostrados en la tabla, es la más representativa. También podemos observar que la existencia de ruido, en funciones con tan bajo porcentaje de aciertos, podría hasta resultar beneficioso, como lo es para la función 4; si usamos la parábola para etiquetar los puntos generados por la distribución uniforme en el intervalo especificado, con semilla de aleatoriedad 1, todos los puntos toman un valor negativo, y dado que el 10% de los puntos positivos de nuestra muestra es superior al 10% de los puntos negativos (hay más puntos por encima de la recta que por debajo), al añadir ruido aumenta el porcentaje de aciertos.



Estas gráficas muestran el porcentaje de aciertos de cada función con ruido. La función menos representativa en este caso es la que utiliza una hipérbola como frontera de clasificación (tercera función).

Ejercicio sobre Modelos Lineales

Apartado 1

Para este apartado se pide implementar la función ajustar_PLA(datos, label, max_iter, inicio) que calcula el hiperplano solución a un problema de clasificación binaria usando el algoritmo Perceptrón

```
Mientras contador sea menor que max_iter:
   para cada elemento de datos:
      si el signo de la matriz transpuesta de inicio por datos[i] es distinto a label[i]
         aumentar inicio en label[i]*datos[i] unidades
   fin para
   si inicio no ha cambiado salir
   aumentar contador
Fin mientras
Devolver inicio y contador
```

Tras inicializar el algoritmo con diferentes puntos de inicio (incluyendo el vector cero) sobre los datos del ejercicio 1 apartado 2 a, se obtienen los siguientes números de iteraciones necesarias para converger:

```
------------------------------ Sin ruido ------------------------------

      Punto inicial                                                                  Iteración
0     [ 0.0, 0.0, 0.0 ]                                                              121
1     [ 0.38403076811573245, 0.2573028872050038, 0.8294019198272359 ]                103
2     [ 0.7363827038385634, 0.5076009080549594, 0.644326615041417 ]                      517
3     [ 0.2131865653786481, 0.8957089487035581, 0.9659462515078151 ]                     13
4     [ 0.31700156230578813, 0.8655526182272831, 0.310283706940119 ]                 436
5     [ 0.025263945431817092, 0.049195157773728626, 0.1846268380142655 ]             96
6     [ 0.06903334161725405, 0.2574754234045431, 0.9135817315706746 ]                514
7     [ 0.4578495119744105, 0.13021178947959489, 0.8098916654001787 ]                    141
8     [ 0.40346984048066603, 0.02443264498316411, 0.8568310426277526 ]               481
9     [ 0.2742948190153962, 0.7091059640719906, 0.3557723428236633 ]                 434
10    [ 0.7943090618265342, 0.8446188611438505, 0.5381475171399585 ]                 111

Nº medio de iteraciones:  269.72727272727275
```

No se han realizado varias pruebas con cada punto inicial puesto que no se trata de un proceso aleatorio. A continuación se pide repetir esto pero variando el conjunto: se realizará sobre los datos del ejercicio 1 apartado 2 b, es decir, se le añadirá ruido a las etiquetas. Se obtienen, con los mismos puntos de inicio, las siguientes iteraciones:

```
------------------------------ Con ruido ------------------------------

      Punto inicial                                                                  Iteración
0     [ 0.0, 0.0, 0.0 ]                                                              5000
1     [ 0.38403076811573245, 0.2573028872050038, 0.8294019198272359 ]                5000
2     [ 0.7363827038385634, 0.5076009080549594, 0.644326615041417 ]                      5000
3     [ 0.2131865653786481, 0.8957089487035581, 0.9659462515078151 ]                     5000
4     [ 0.31700156230578813, 0.8655526182272831, 0.310283706940119 ]                 5000
5     [ 0.025263945431817092, 0.049195157773728626, 0.1846268380142655 ]             5000
6     [ 0.06903334161725405, 0.2574754234045431, 0.9135817315706746 ]                5000
7     [ 0.4578495119744105, 0.13021178947959489, 0.8098916654001787 ]                    5000
```

```
8       [ 0.40346984048066603, 0.02443264498316411, 0.8568310426277526 ]    5000
9       [ 0.2742948190153962, 0.7091059640719906, 0.3557723428236633 ]      5000
10      [ 0.7943090618265342, 0.8446188611438505, 0.5381475171399585 ]      5000

Nº medio de iteraciones:  5000.0
```

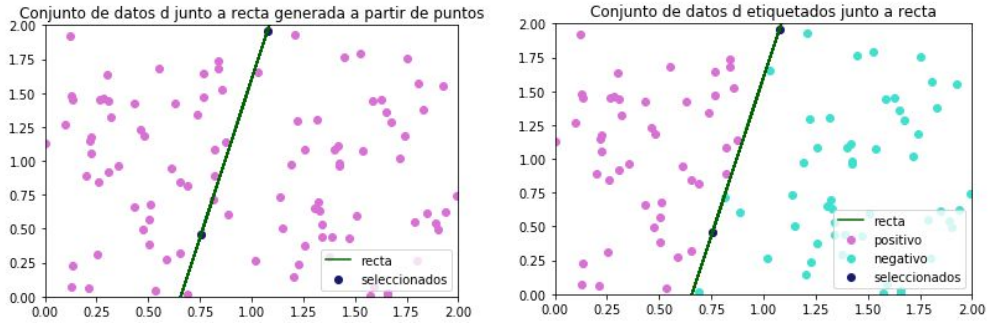
Como se puede observar, al añadirle ruido a las etiquetas, el número de iteraciones coincide con el máximo de iteraciones asignado (5000), es decir, este máximo no resulta suficiente para que el algoritmo converja partiendo de los puntos iniciales establecidos. Este resultados tienen sentido puesto que al existir ruido, nunca se dará la condición de parada (la variable inicio no ha cambiado su valor en la iteración actual).

Apartado 2

Para este apartado se pide implementar la función sgd_rl(x,etiquetas,w)

```
Mientras diferencia (inf) sea menor que diferencia(0,01):
   desordenar x y etiquetas
   aux es w
   para cada elemento de x y etiquetas:
      aumentar gradiente en
          e^(y*matriz transpuesta de w * x)
          ---------------------------------
          1 + e^(y*transpuesta de w * x)
   fin para
   dividir -gradiente entre el número de datos
   reducir w en learning_rate(0,01)*gradiente unidades
   diferencia es aux - w
Fin mientras
Devolver w
```

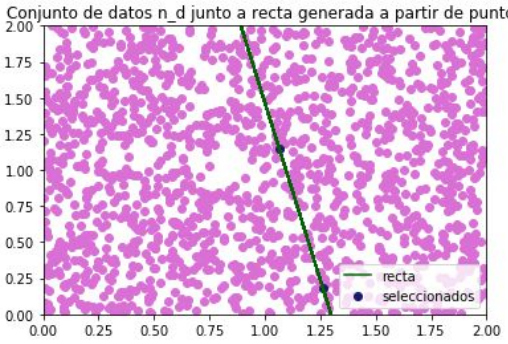
Estudiaremos el funcionamiento de la regresión logística con gradiente descendente estocástico sobre una muestra de 100 puntos de 2 dimensiones distribuidos uniformemente en el intervalo [0,2]. De ellos se han seleccionado aleatoriamente dos puntos, por los cuales pasará la recta que usaremos para la clasificación de los puntos

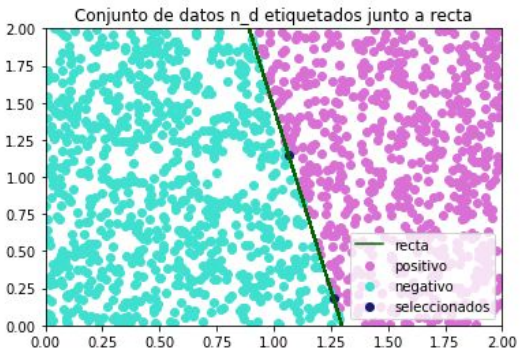


Sobre este conjunto aplicamos el algoritmo y evaluamos la bondad de la solución obtenida empleando el riesgo del error empírico

```
Error de regresión logística en la muestra:  0.6931471805599453
```

Tras esto, se pide volver a aplicar el algoritmo sobre una muestra de tamaño mayor a 999. Para ello, repetimos el proceso de generación de 1500 puntos, selección de dos de ellos y clasificación en función a la recta generada por los puntos seleccionados.





Error de regresión logística en la nueva muestra: 0.6931470601797215

Bibliografía

- https://docs.scipy.org
- http://www.sc.ehu.es
- https://es.wikipedia.org
- https://medium.com
- http://towardsdatascience.com