



# Tarea 3: Métodos No Lineales



Juan Pablo Castillo  
Álvaro Rojas



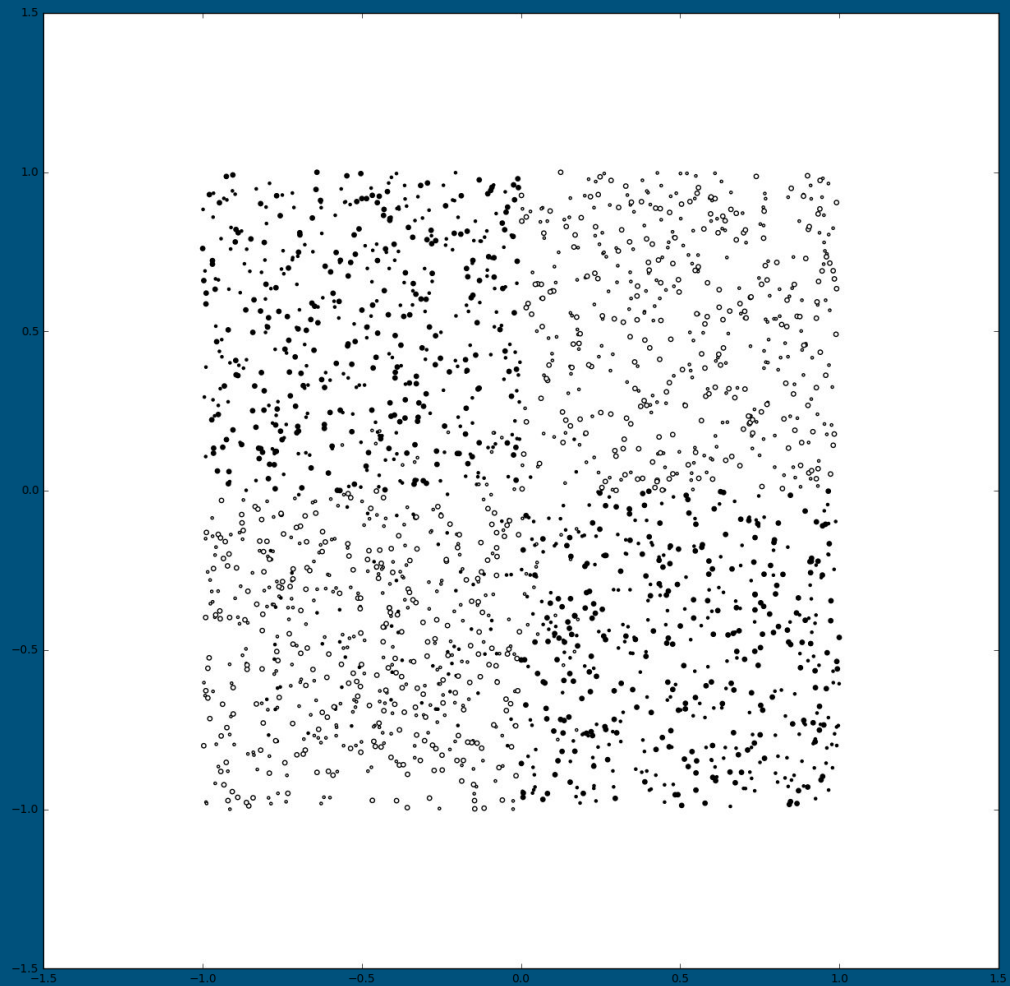
---

# 1. El Viejo XOR: Métodos No-lineales para Problemas No-lineales

# Contexto

---

En esta sección se tiene el objetivo de experimentar con algunos modelos no-lineales sobre un problema de juguete, el cual es bastante famoso en la historia del aprendizaje automático, este es el problema XOR. Se trata de un problema de clasificación linealmente inseparable. En este problema se utilizará distintos modelos no-lineales para analizar su efectividad en clasificar las clases en este problema.



# Contexto

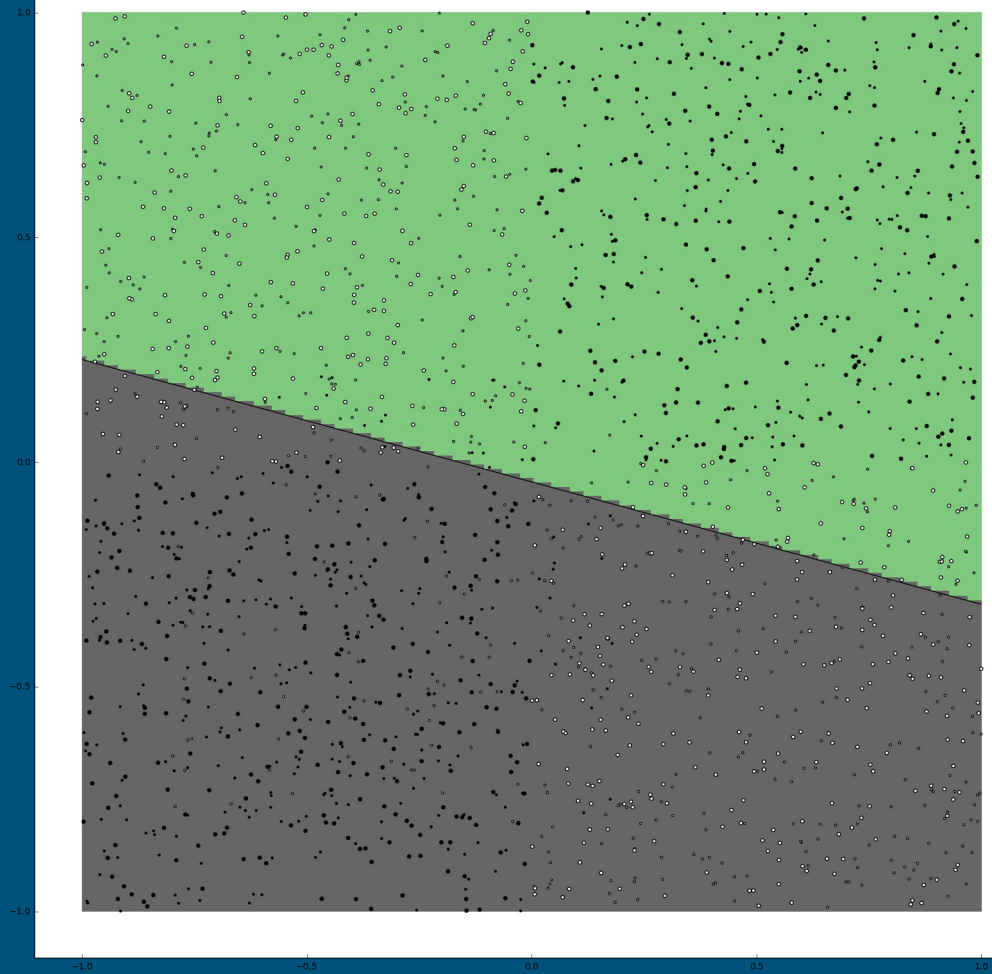
---

- Una clase se encuentra en el cuadrante 1 y 3, mientras que la otra clase se encuentra en el cuadrante 2 y 4.
- Es similar al concepto de un "or exclusivo", donde se tiene un valor 1 sólo cuando  $x$  e  $y$  son distintos. Pero si son iguales, se tendrá un valor 0.

# Clasificación con una SVM lineal

```
Mejor parámetro de regularización C: 0.01
Mejor Accuracy de Test: 0.471000
Training Accuracy SVM Lineal: 0.517273
Test Accuracy SVM Lineal: 0.471000
Detailed Analysis Testing Results ...
```

	precision	recall	f1-score	support
-1	0.46	0.47	0.46	489
+1	0.48	0.48	0.48	511
avg / total	0.47	0.47	0.47	1000



—

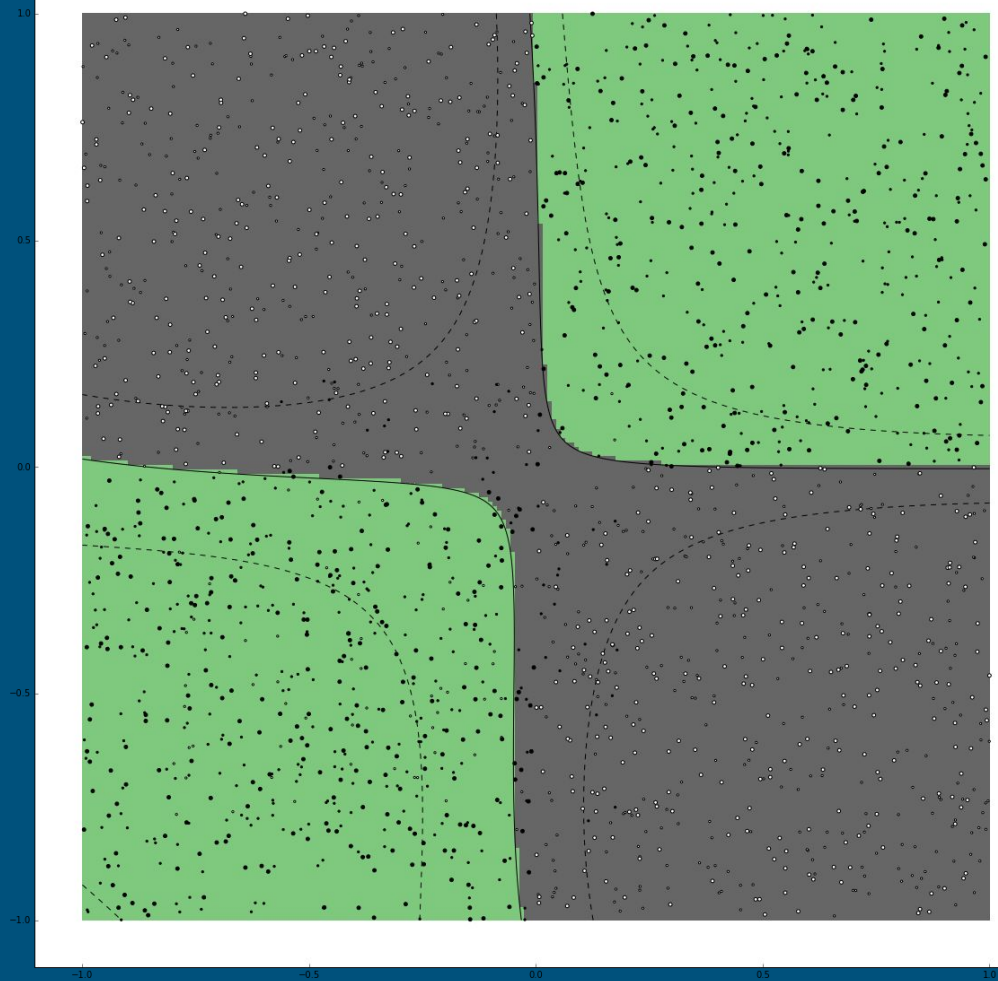
# Clasificación con SVM no-lineales (rbf)

```
Mejor parámetro de regularización C: 8
Mejor Accuracy de Test: 0.972000
Training Accuracy SVM con kernel rbf: 0.902727
Test Accuracy SVM con kernel rbf: 0.972000
Detailed Analysis Testing Results ...
```

	precision	recall	f1-score	support
-1	0.99	0.95	0.97	489
+1	0.95	0.99	0.97	511
avg / total	0.97	0.97	0.97	1000



—

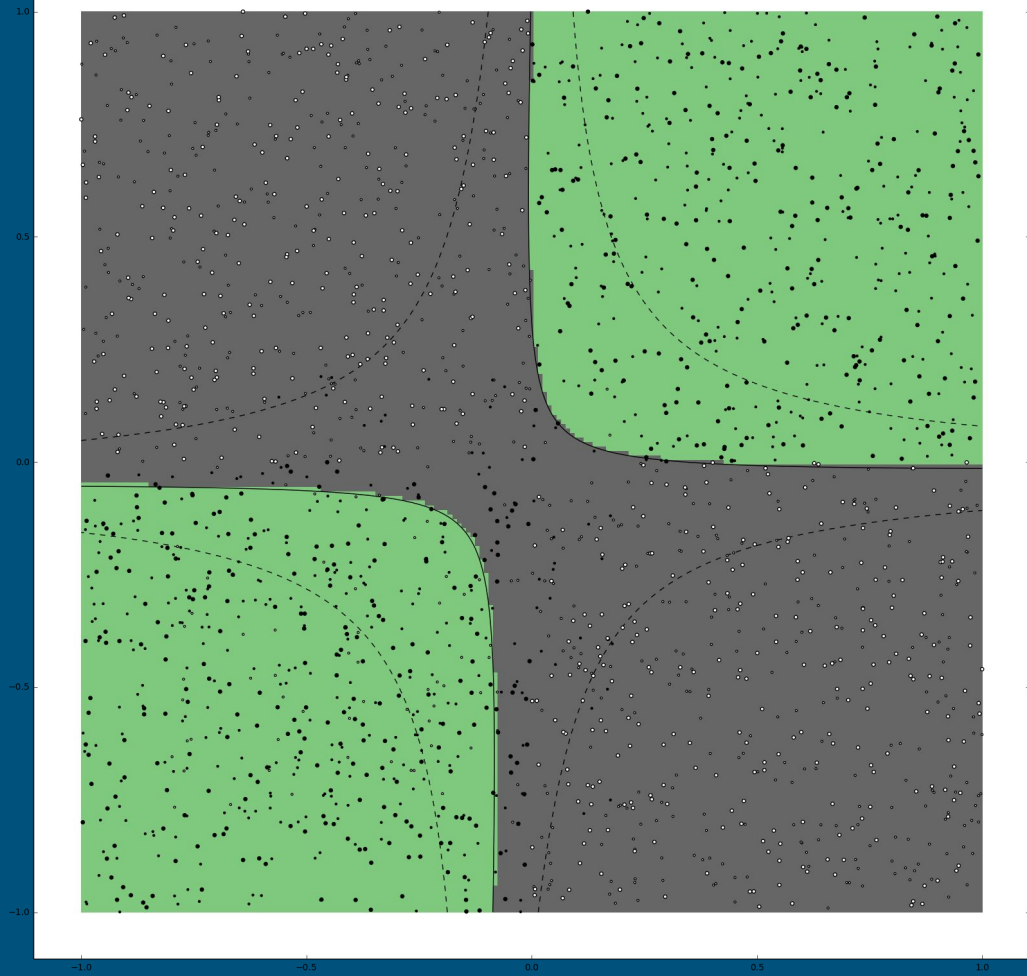


# Clasificación con SVM no-lineales (polynomial)

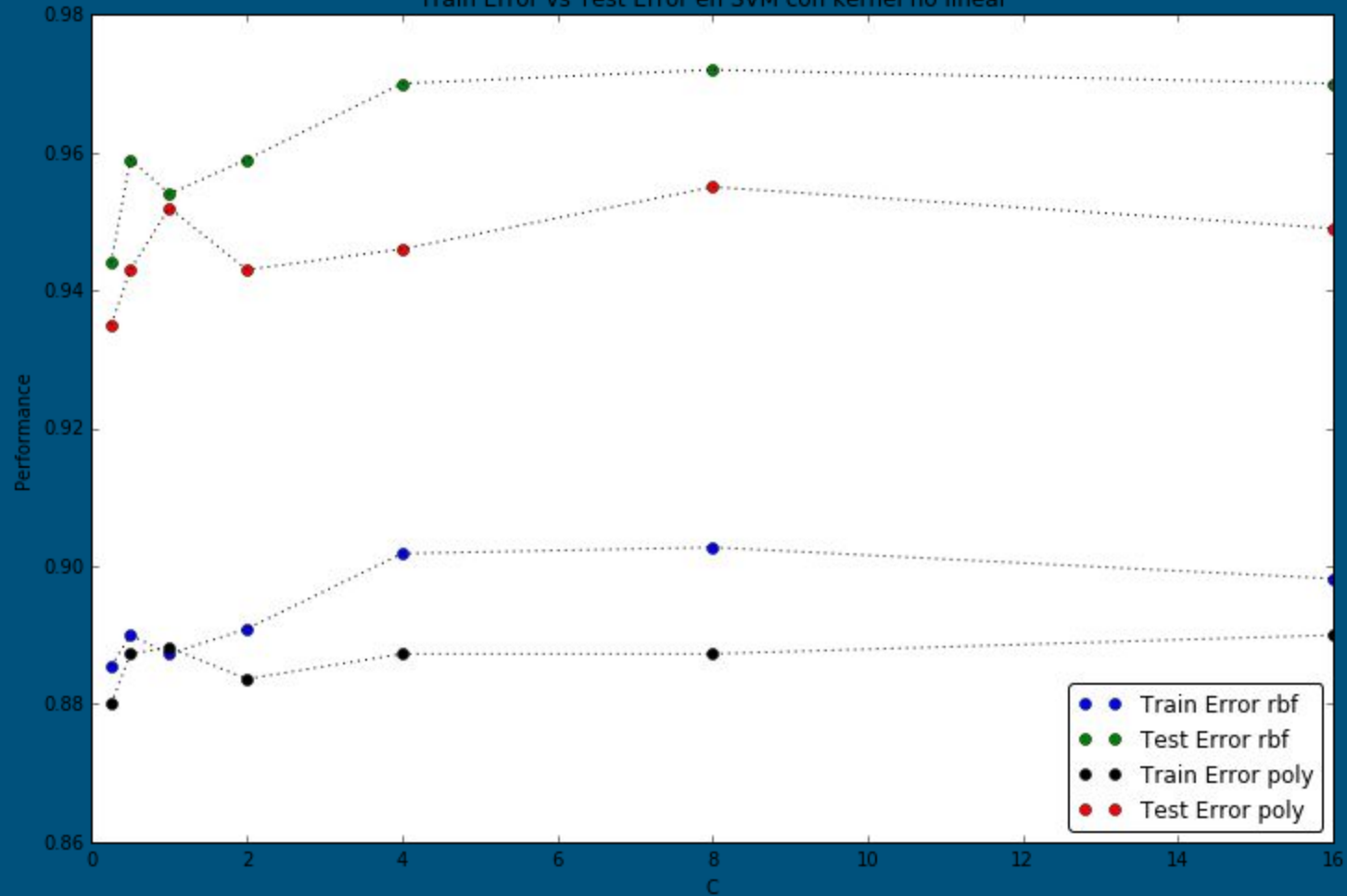
```
Mejor parámetro de regularización C: 8
Mejor Accuracy de Test: 0.955000
Training Accuracy SVM con kernel Polinomial: 0.887273
Test Accuracy SVM con kernel Polinomial: 0.955000
Detailed Analysis Testing Results ...
```

	precision	recall	f1-score	support
-1	0.99	0.92	0.95	489
+1	0.93	0.99	0.96	511
avg / total	0.96	0.95	0.95	1000

—

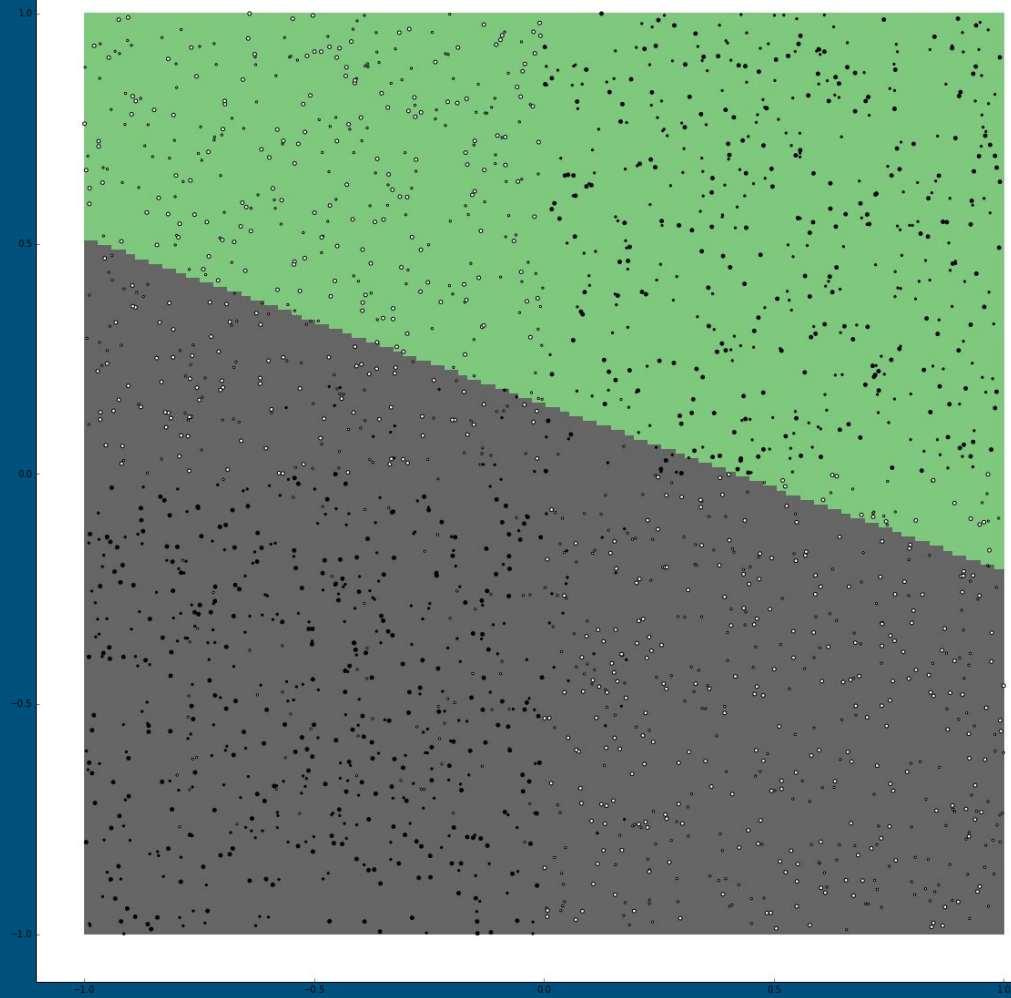


Train Error vs Test Error en SVM con kernel no lineal



# Clasificación con red neuronal artificial de 1 sola neurona

```
Entrenando la red neuronal de 1 neurona...  
928/1000 [=====>...] - ETA: 0s  
Test Accuracy = 0.553000
```



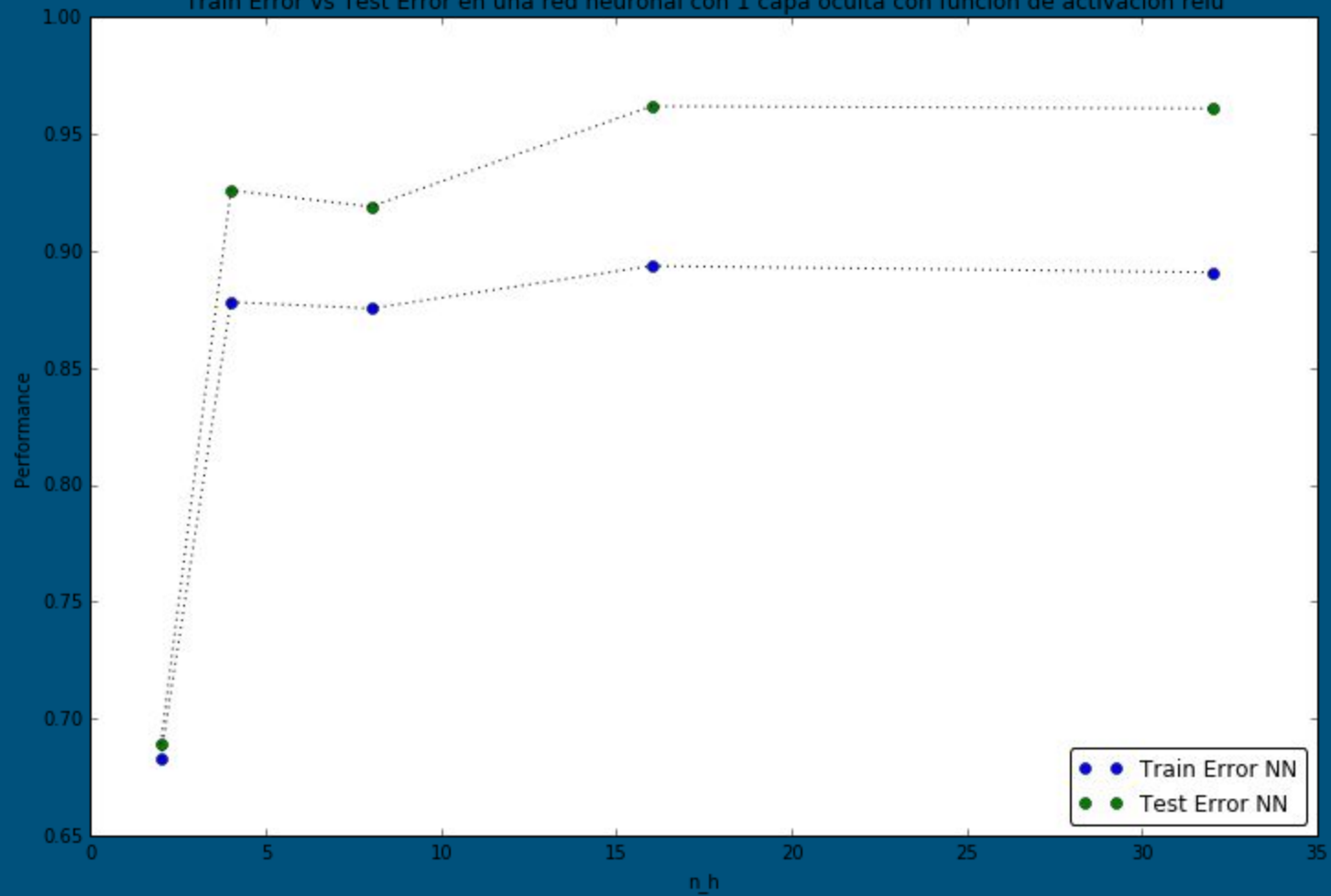
1.0

# Clasificación con una red neuronal artificial de 1 capa escondida

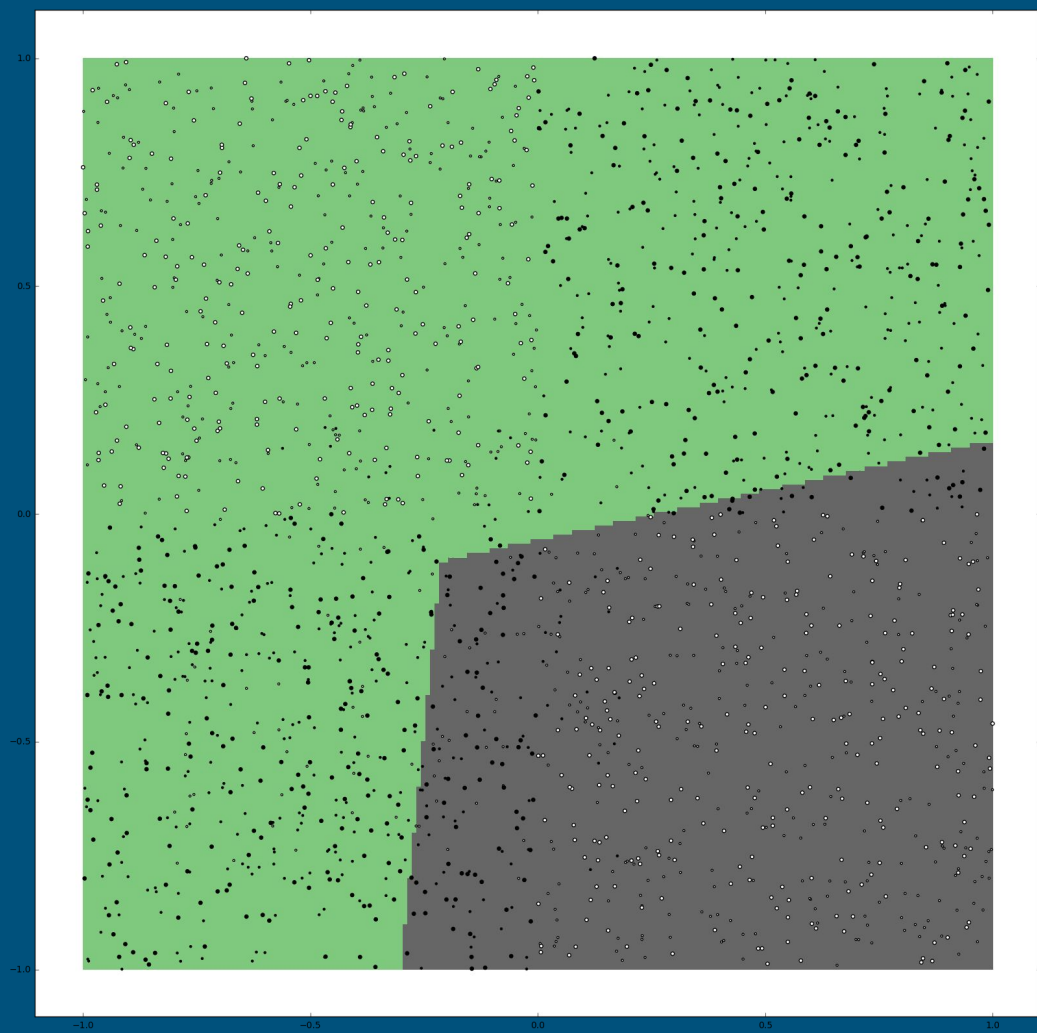
```
992/1000 [=====>.] - ETA: 0s
Test Accuracy con una capa oculta de 2 neuronas = 0.689000
992/1000 [=====>.] - ETA: 0s
Test Accuracy con una capa oculta de 4 neuronas = 0.926000
1000/1000 [=====] - 0s

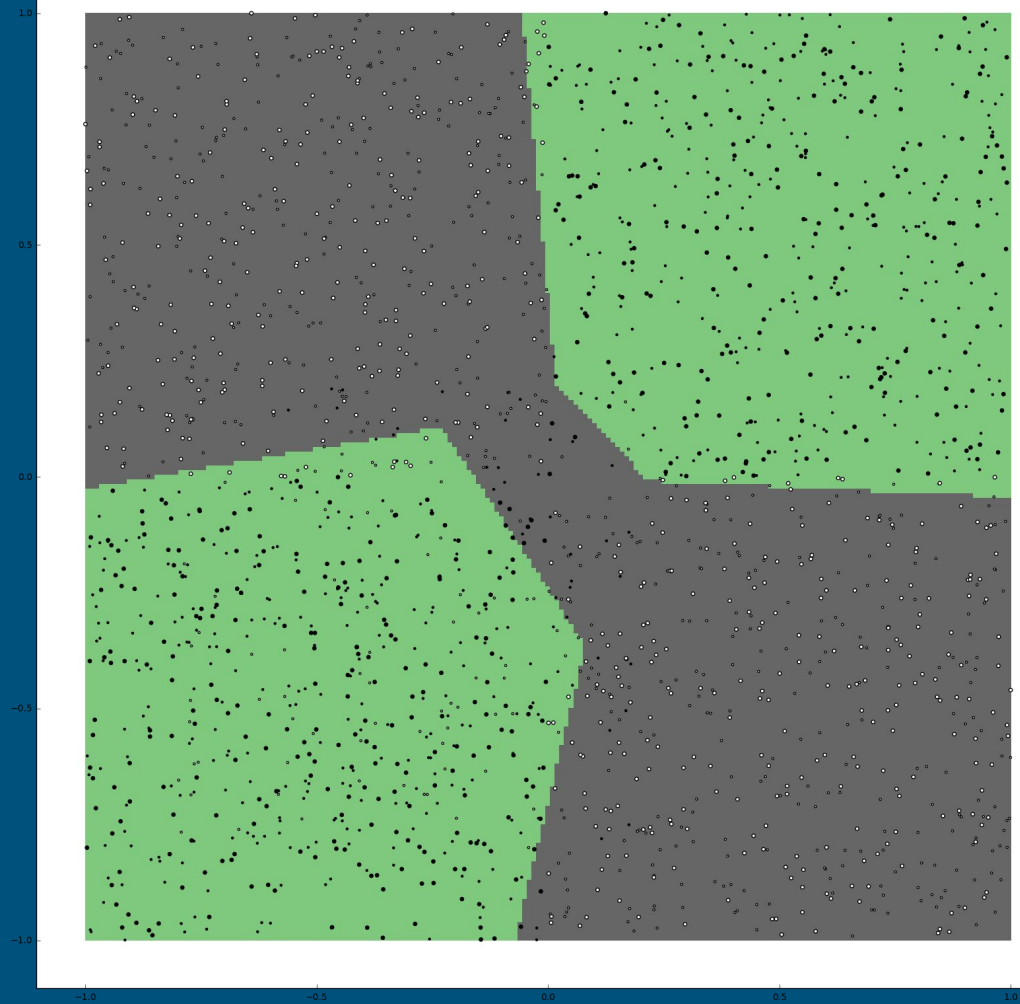
Test Accuracy con una capa oculta de 8 neuronas = 0.919000
992/1000 [=====>.] - ETA: 0s
Test Accuracy con una capa oculta de 16 neuronas = 0.962000
1100/1100 [=====] - 0s
992/1000 [=====>.] - ETA: 0s
Test Accuracy con una capa oculta de 32 neuronas = 0.961000
```

Train Error vs Test Error en una red neuronal con 1 capa oculta con funcion de activacion relu









1.0

0.5

0.0

-0.5

-1.0

-1.0

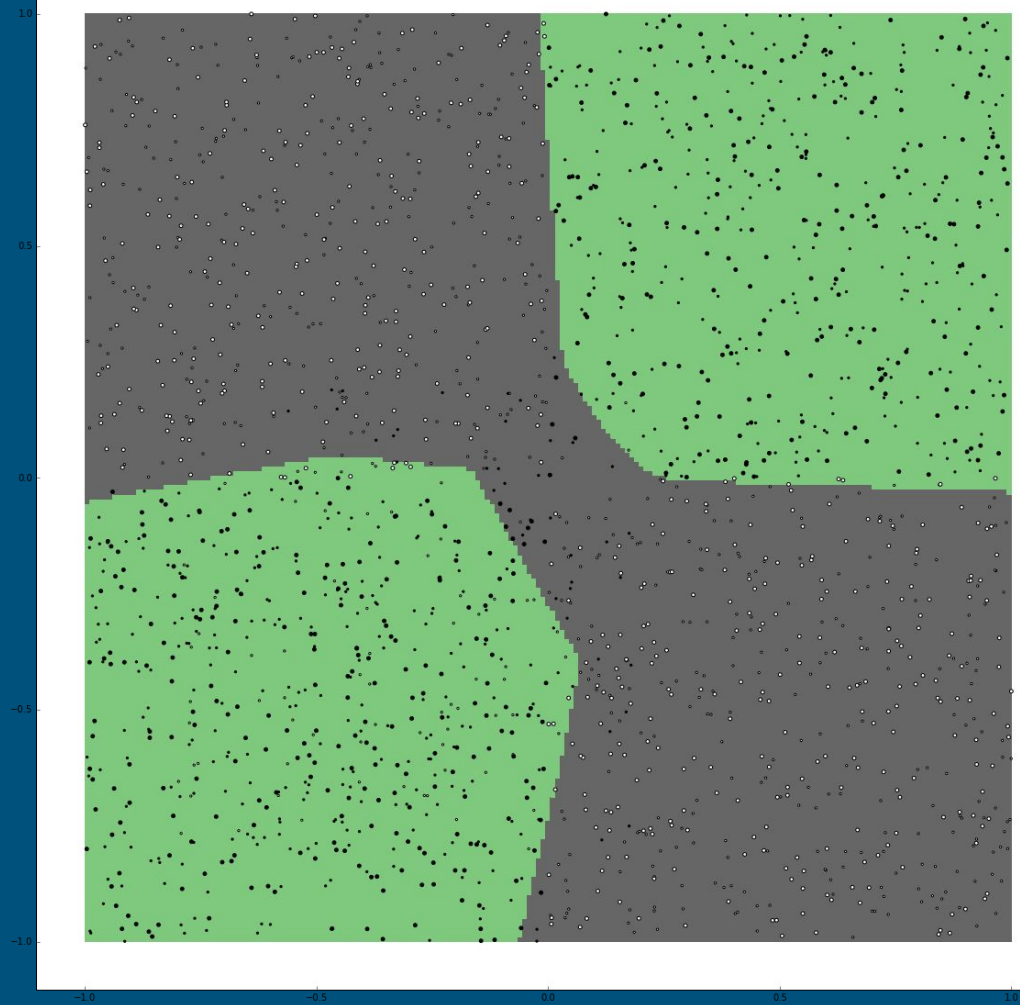
-0.5

0.0

0.5

1.0

—



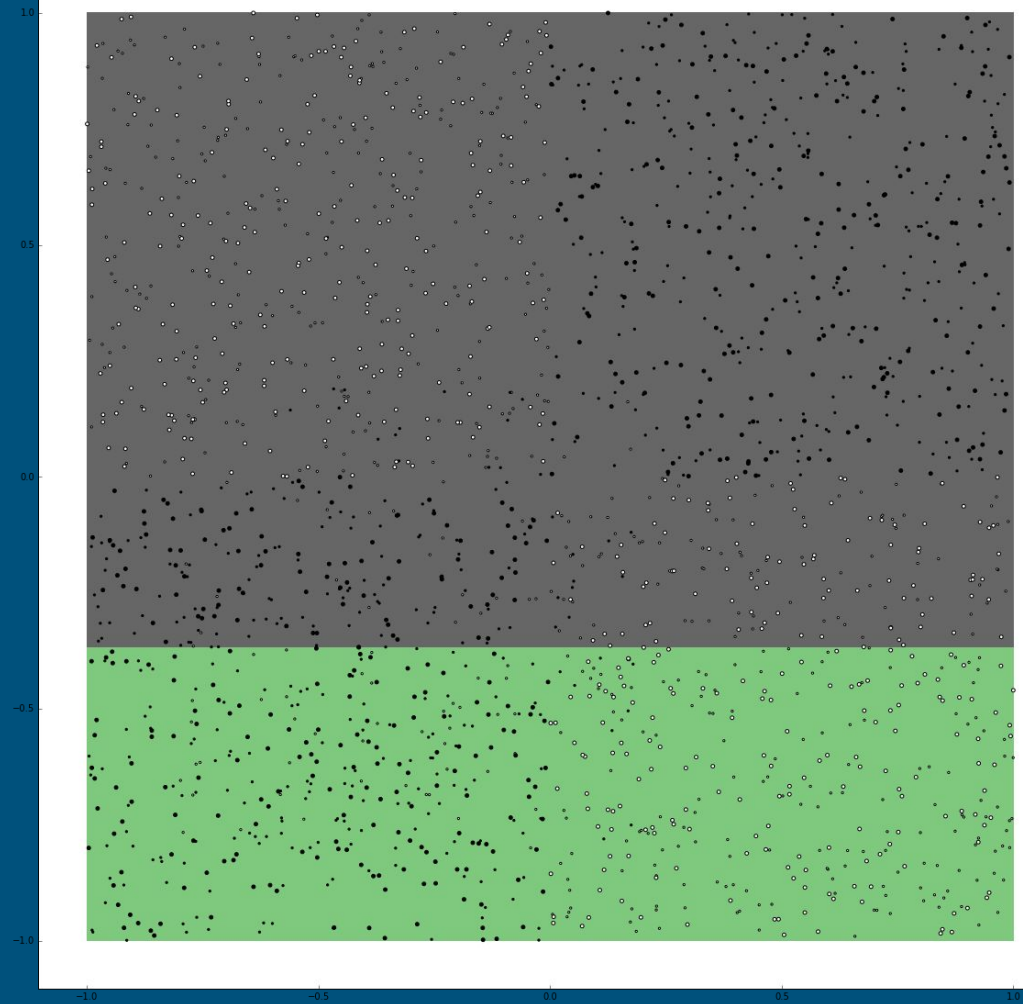
# Clasificación con un árbol de 1 nivel

---

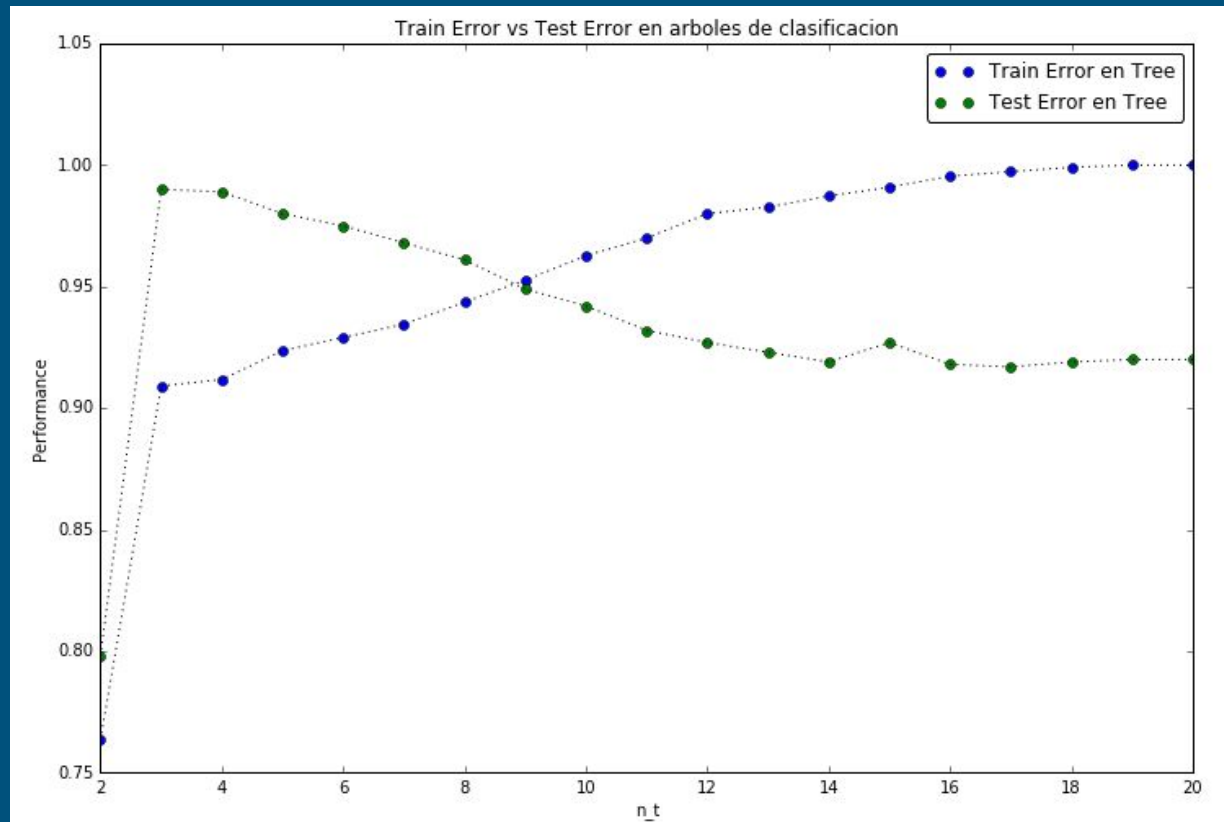
```
Training Accuracy Arbol con 1 nivel: 0.541818
Test Accuracy Arbol con 1 nivel: 0.489000
Detailed Analysis Testing Results ...
```

	precision	recall	f1-score	support
-1	0.48	0.67	0.56	489
+1	0.50	0.32	0.39	511
avg / total	0.49	0.49	0.47	1000

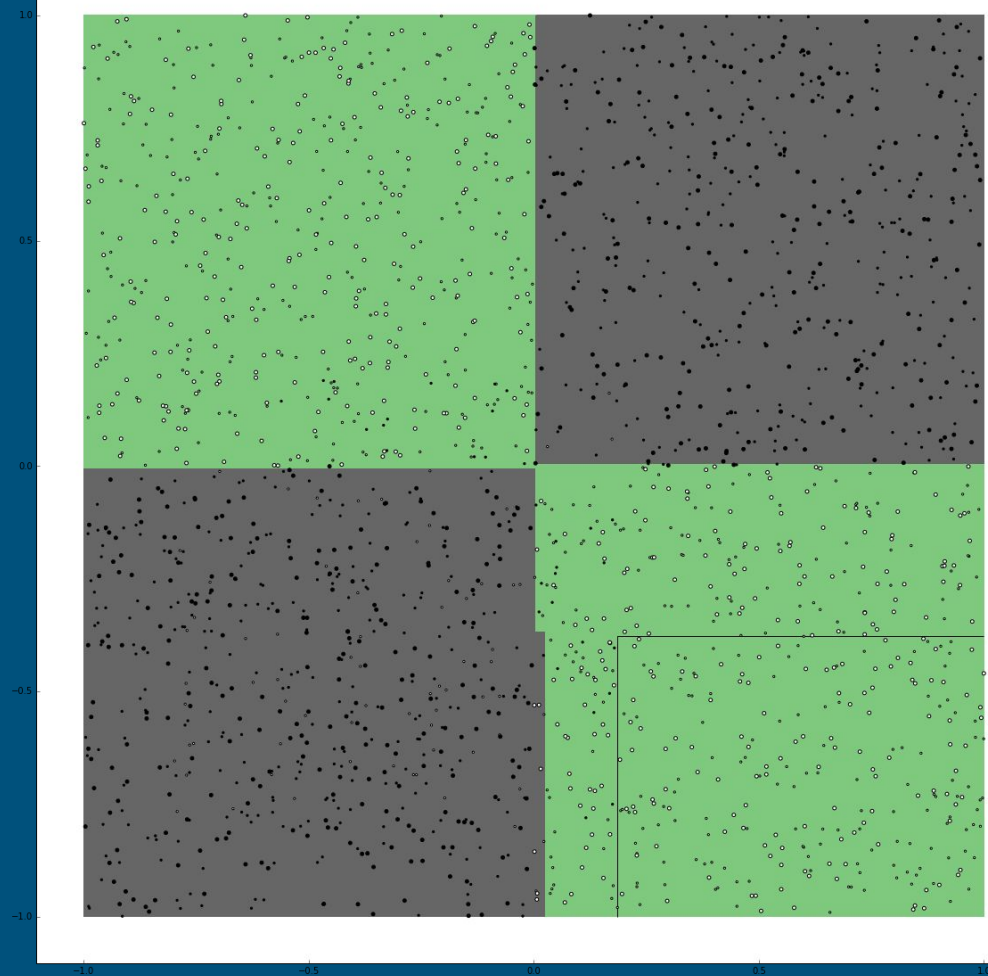
Test Accuracy = 0.489000  
1



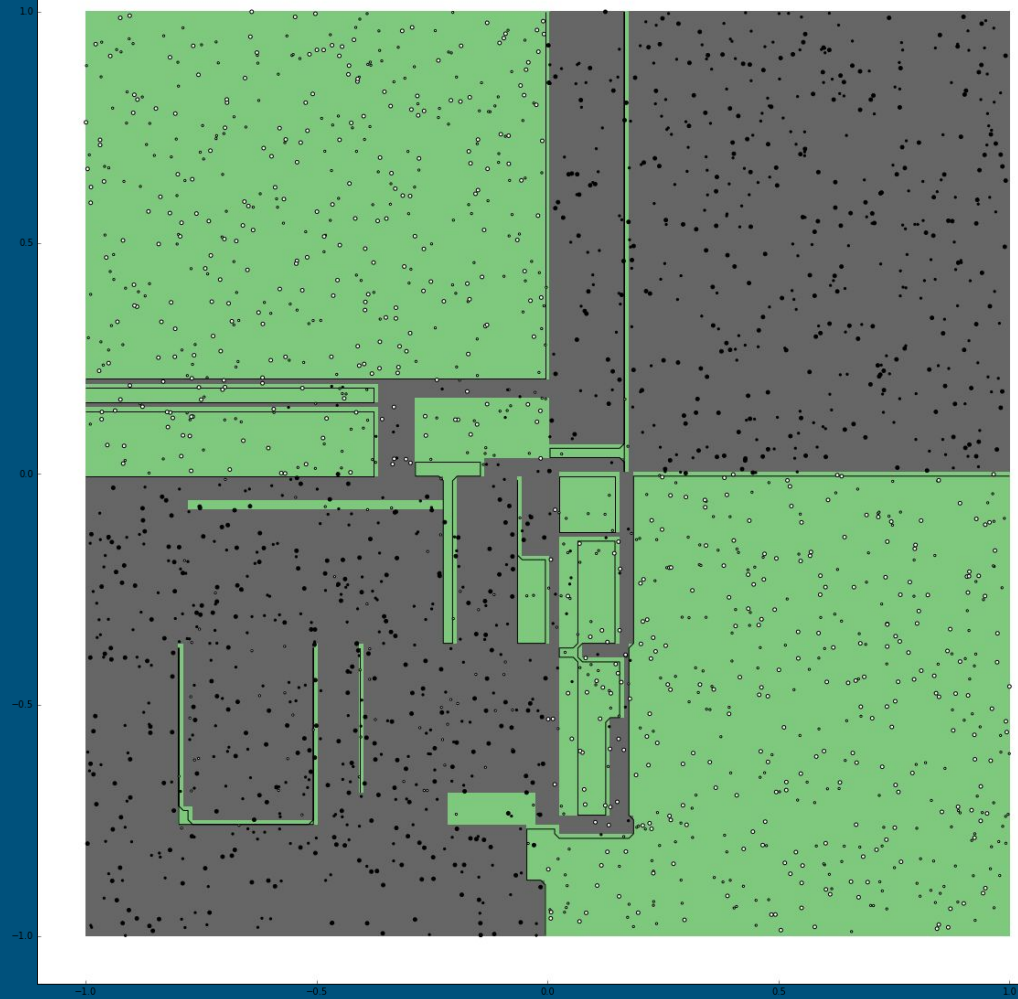
# Clasificación con un árbol de múltiples niveles



Clasificador Tree con  $n_t = 3$   
Test Accuracy = 0.990000

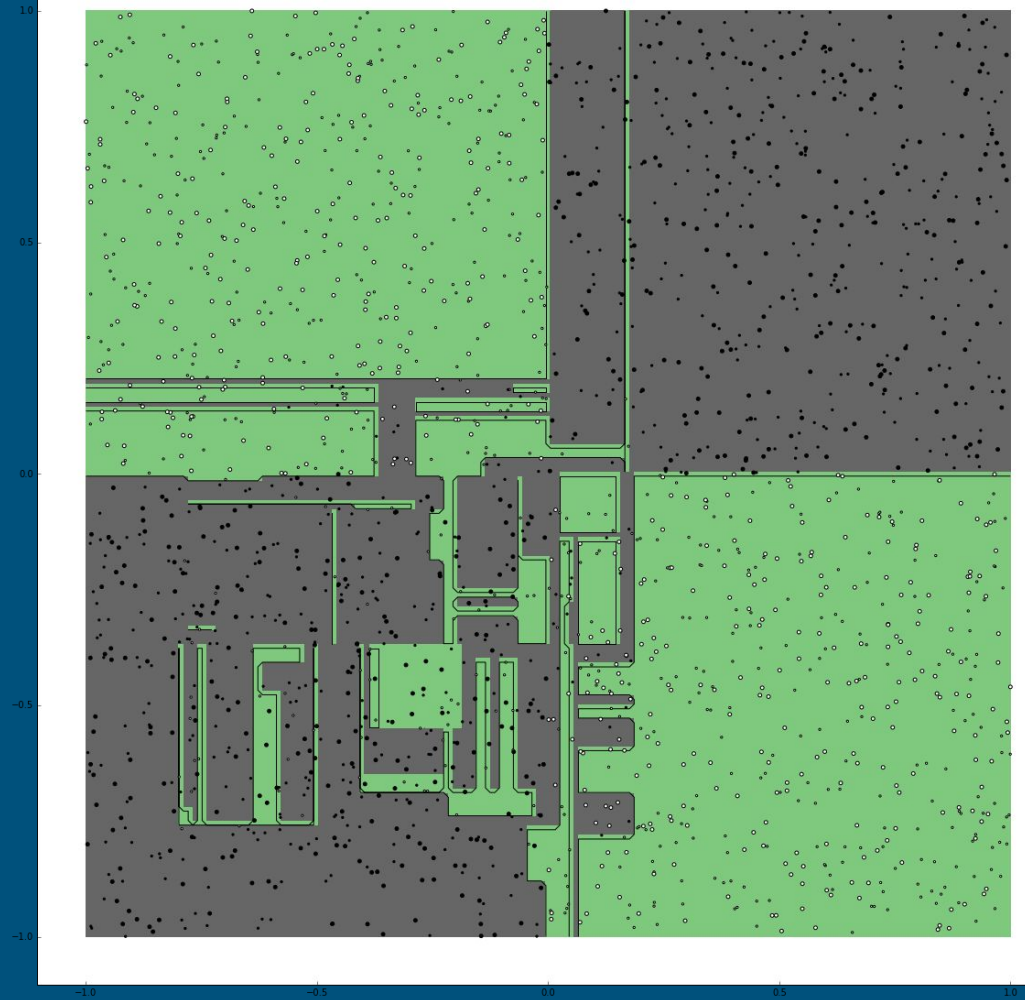


Clasificador Tree con  $n_t = 9$   
Test Accuracy = 0.949000





Clasificador Tree con  $n_t = 15$   
Test Accuracy = 0.927000



---

## 2. Bike Sharing: Predicción de Demanda Horaria

# Contexto

---

- Los sistemas de intercambio de bicicletas son un medio para alquilar bicicletas en las que el proceso de obtención de membresía, alquiler y retorno de bicicletas se automatiza a través de una red de ubicaciones de quioscos en toda la ciudad.
- Utilizando estos sistemas, la gente puede alquilar una bicicleta desde una ubicación y devolverla a un lugar diferente según sea necesario.
- Se pide a los participantes que combinen los patrones de uso histórico con los datos meteorológicos para predecir la demanda de alquiler de bicicletas en el programa Capital Bikeshare en Washington, DC.

# Contexto

Atributo	Descripción
datetime	hourly date + timestamp
season	1 = spring, 2 = summer, 3 = fall, 4 = winter
holiday	whether the day is considered a holiday
workingday	whether the day is neither a weekend nor holiday
weather	1: Clear, Few clouds, Partly cloudy, Partly cloudy 2: Mist + Cloudy, Mist + Broken clouds, Mist + Few clouds, Mist 3: Light Snow, Light Rain + Thunderstorm + Scattered clouds, Light Rain + Scattered clouds 4: Heavy Rain + Ice Pallets + Thunderstorm + Mist, Snow + Fog
temp	temperature in Celsius
atemp	“feels like” temperature in Celsius
humidity	relative humidity
windspeed	wind speed
casual	number of non-registered user rentals initiated
registered	number of registered user rentals initiated
count	number of total rentals

# Contexto

---

Se desea obtener el mejor puntaje posible para la función proporcionada para el concurso:

$$E_{bikes}(y, \hat{y}) = \frac{1}{x} \sum_i (\ln(y_i + 1) - \ln(\hat{y}_i + 1))^2$$

En donde  $y$  e  $\hat{y}$  son los valores reales y de predicción respectivamente.

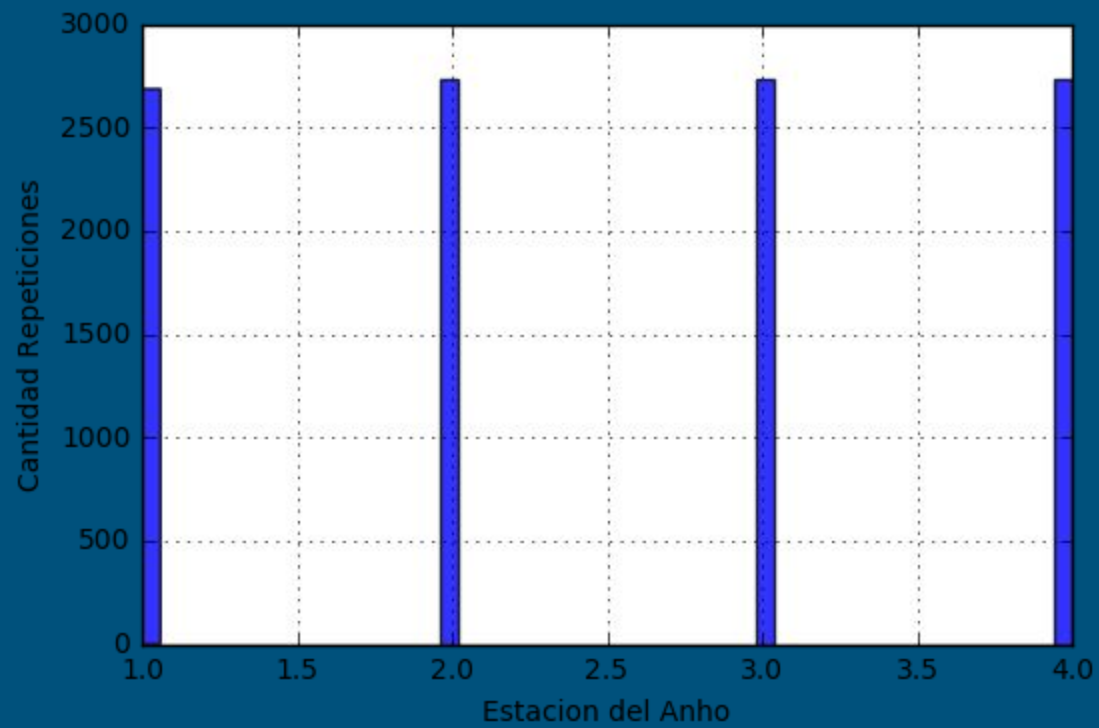
Se construye un dataframe extrayendo el valor de la hora del día desde los datos de fecha y hora de registro para incorporarlos como característica numérica al problema.

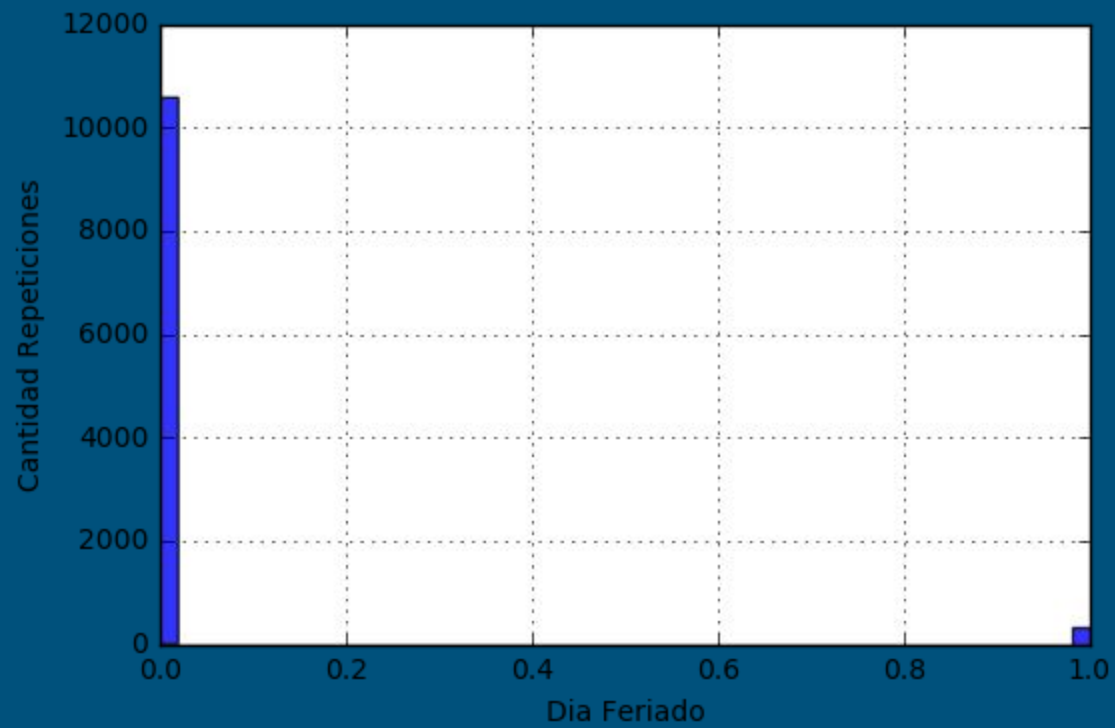
Summary - dataframe completo:

	Unnamed: 0	season	holiday	workingday	weather \
count	10886.000000	10886.000000	10886.000000	10886.000000	10886.000000
mean	5442.500000	2.506614	0.028569	0.680875	1.418427
std	3142.661849	1.116174	0.166599	0.466159	0.633839
min	0.000000	1.000000	0.000000	0.000000	1.000000
25%	2721.250000	2.000000	0.000000	0.000000	1.000000
50%	5442.500000	3.000000	0.000000	1.000000	1.000000
75%	8163.750000	4.000000	0.000000	1.000000	2.000000
max	10885.000000	4.000000	1.000000	1.000000	4.000000

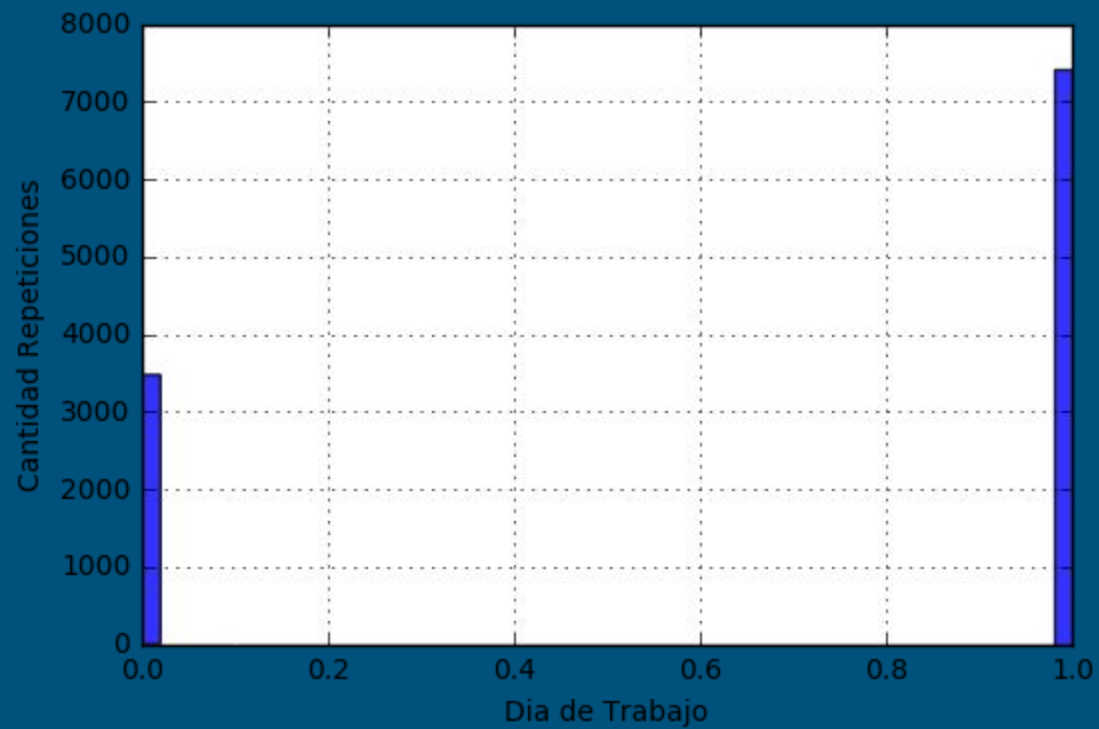
	temp	atemp	humidity	windspeed	casual \
count	10886.000000	10886.000000	10886.000000	10886.000000	10886.000000
mean	20.23086	23.655084	61.886460	12.799395	36.021955
std	7.79159	8.474601	19.245033	8.164537	49.960477
min	0.82000	0.760000	0.000000	0.000000	0.000000
25%	13.94000	16.665000	47.000000	7.001500	4.000000
50%	20.50000	24.240000	62.000000	12.998000	17.000000
75%	26.24000	31.060000	77.000000	16.997900	49.000000
max	41.00000	45.455000	100.000000	56.996900	367.000000

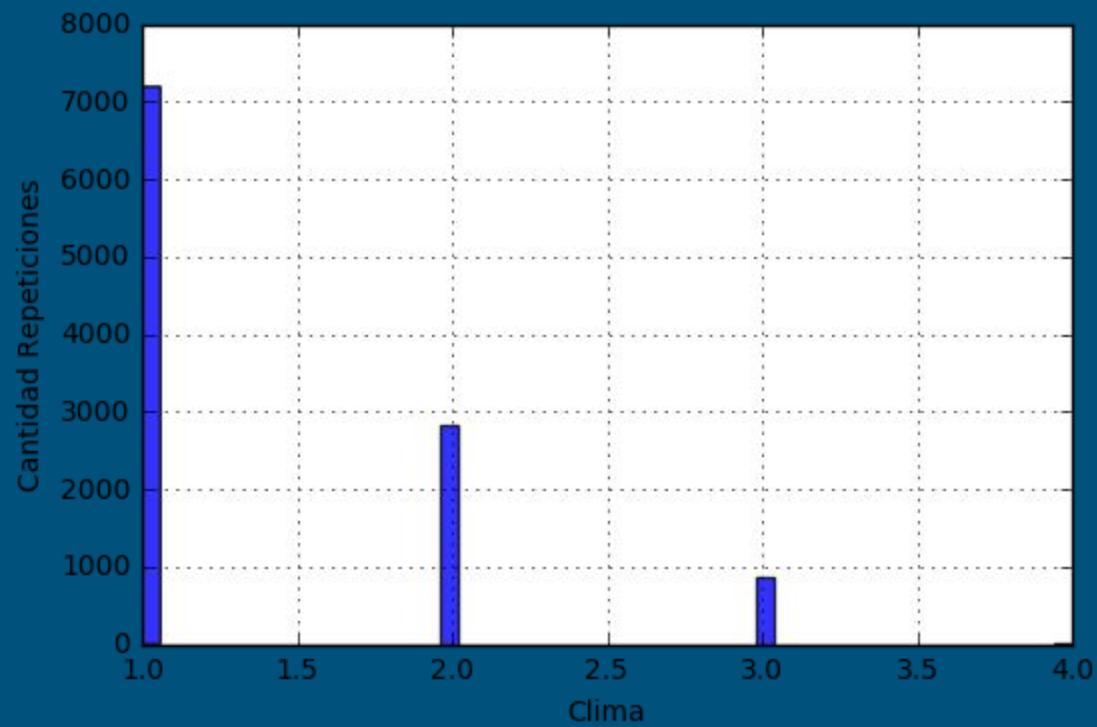
	registered	count
count	10886.000000	10886.000000
mean	155.552177	191.574132
std	151.039033	181.144454
min	0.000000	1.000000
25%	36.000000	42.000000
50%	118.000000	145.000000
75%	222.000000	284.000000
max	886.000000	977.000000

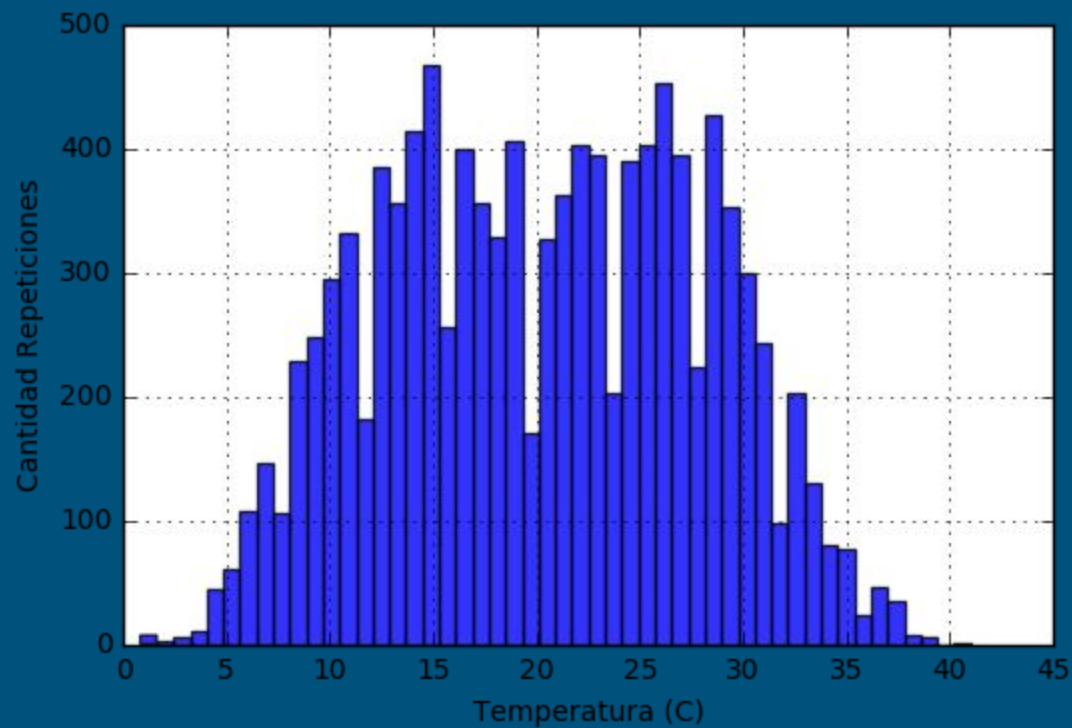


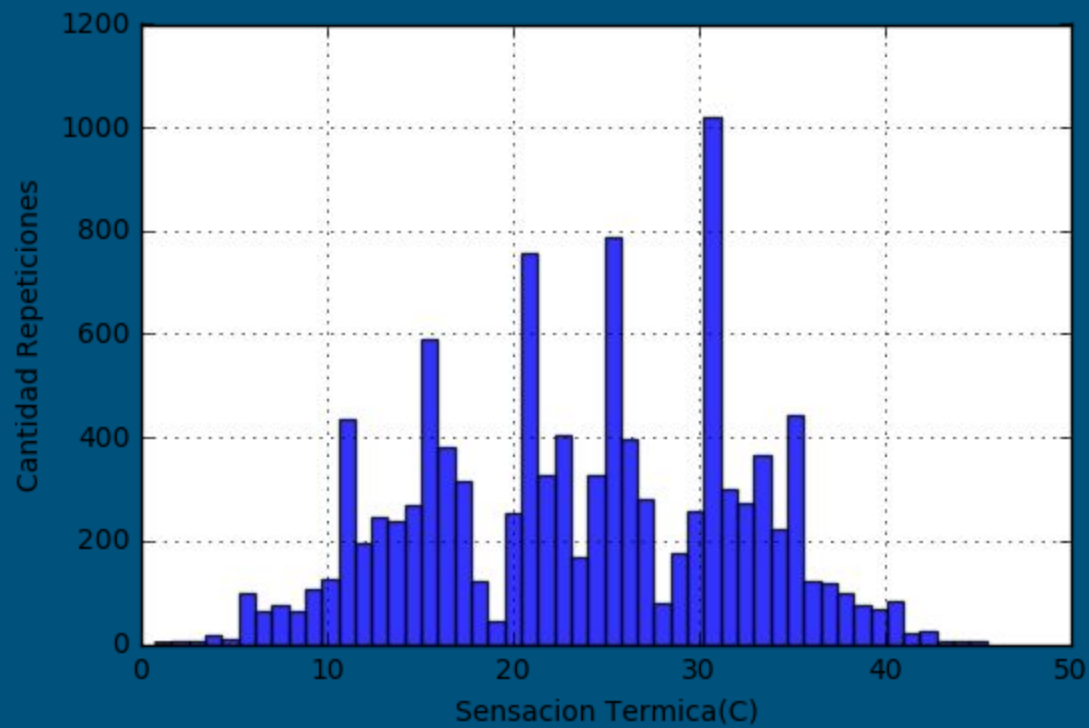


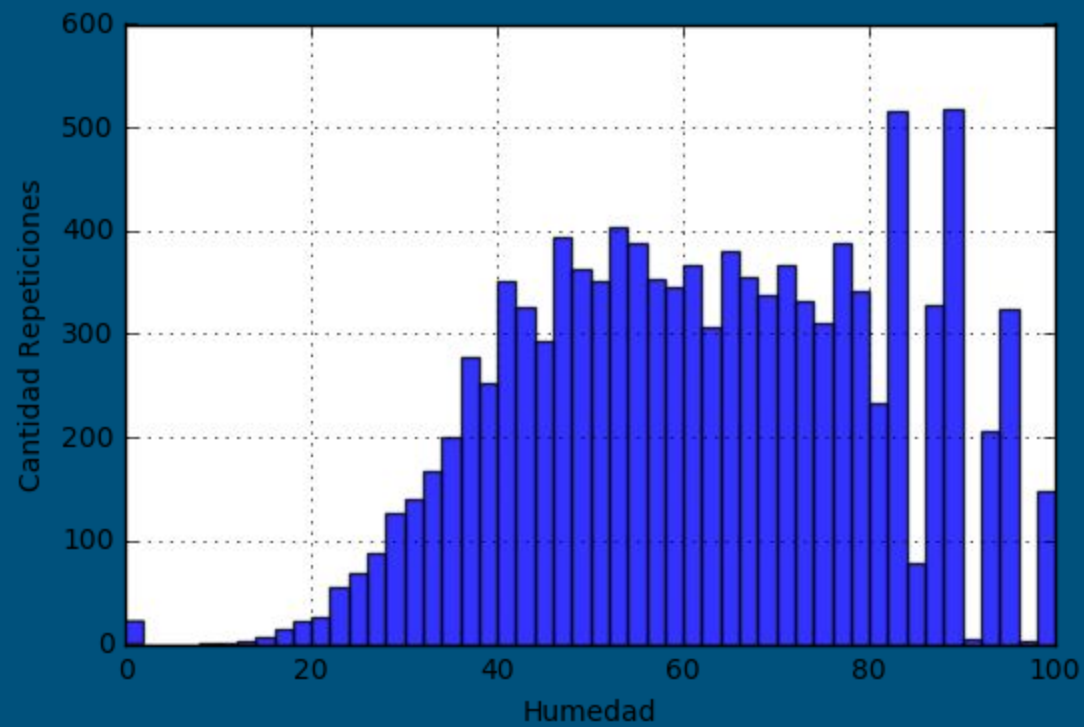


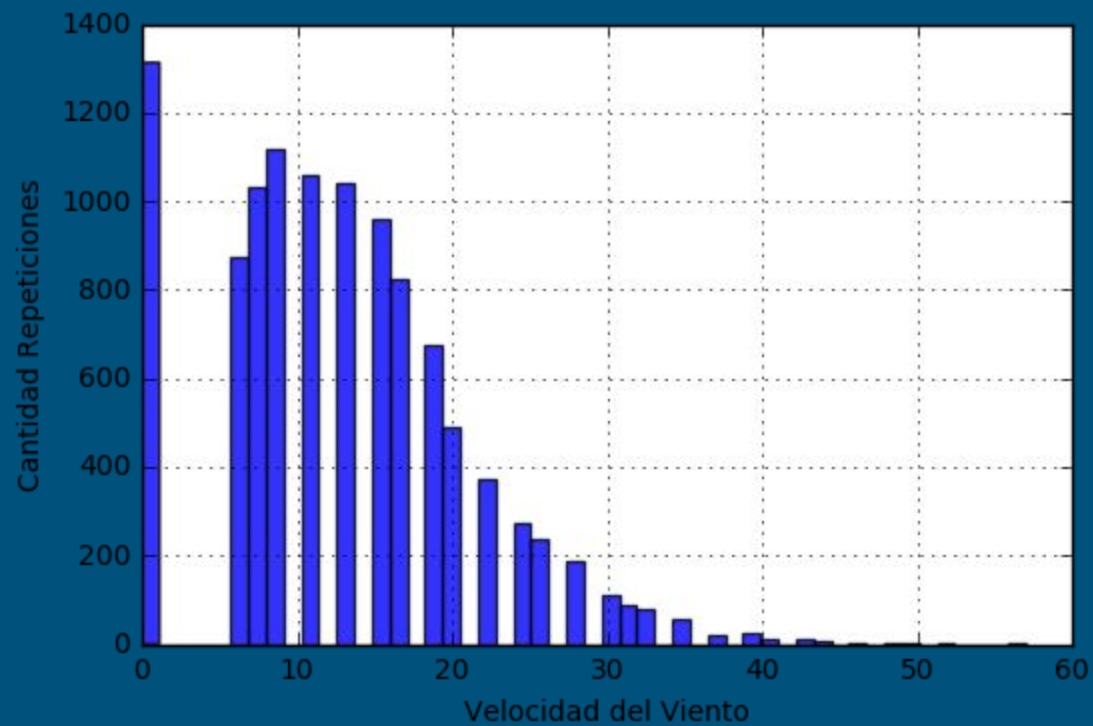


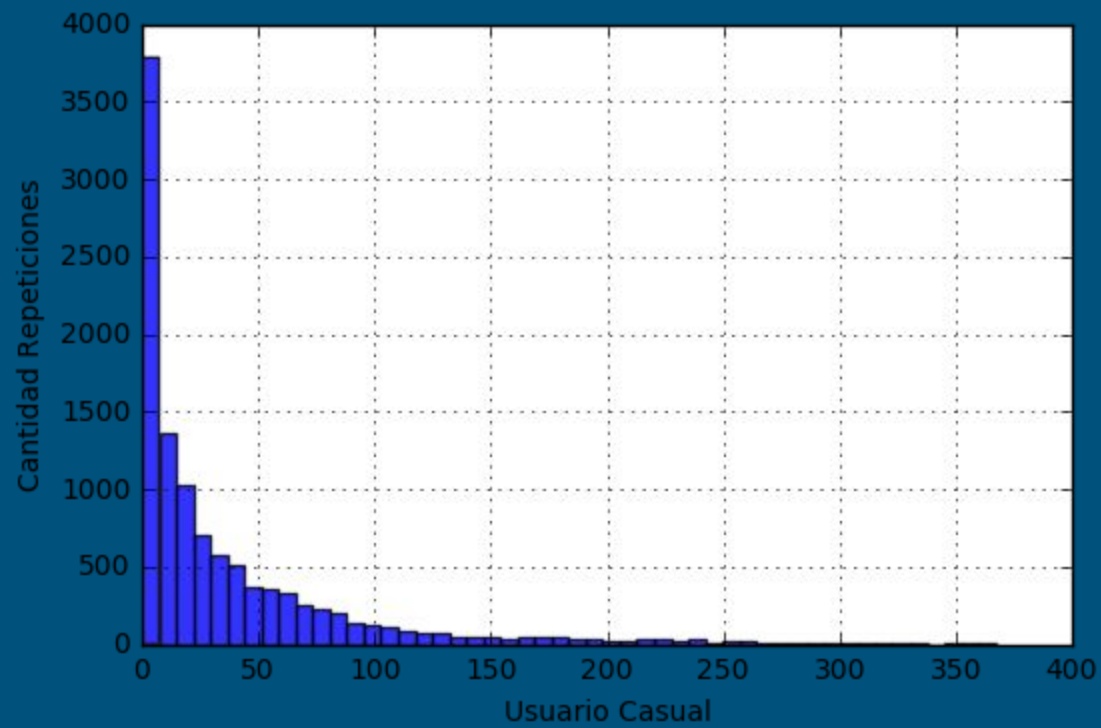


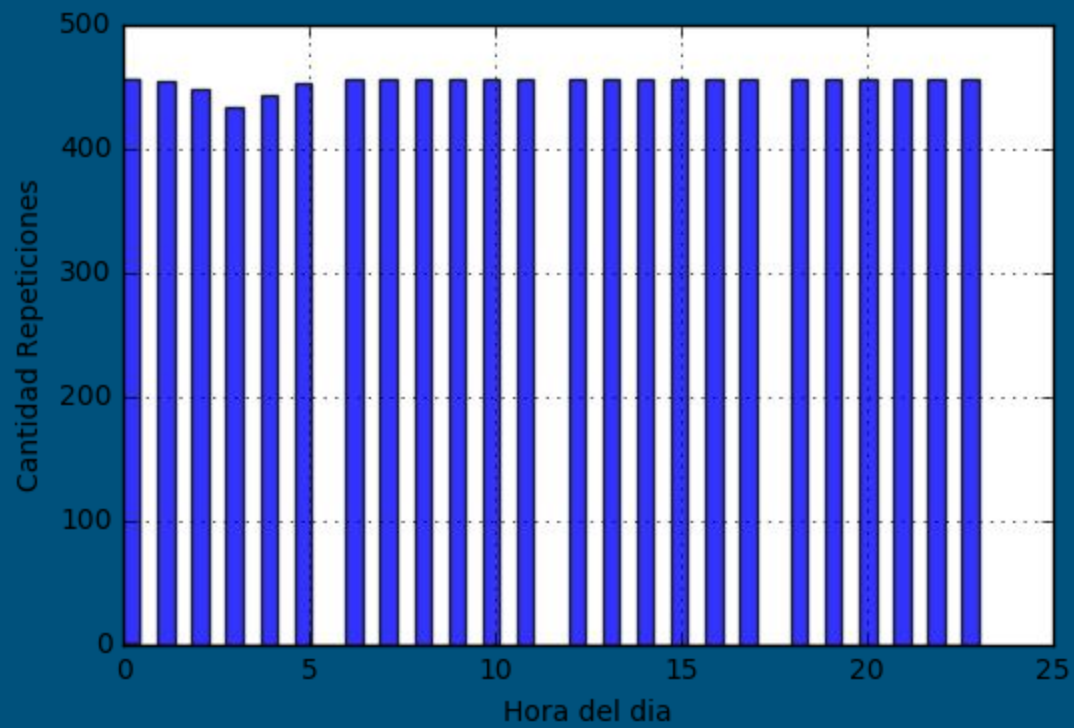




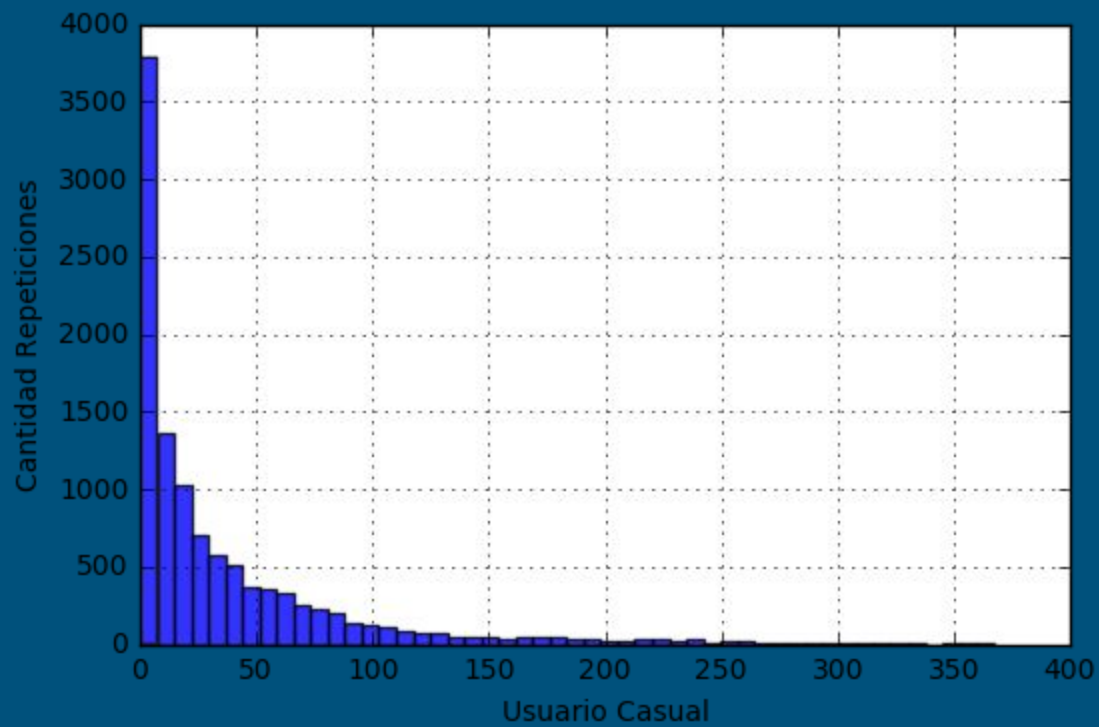


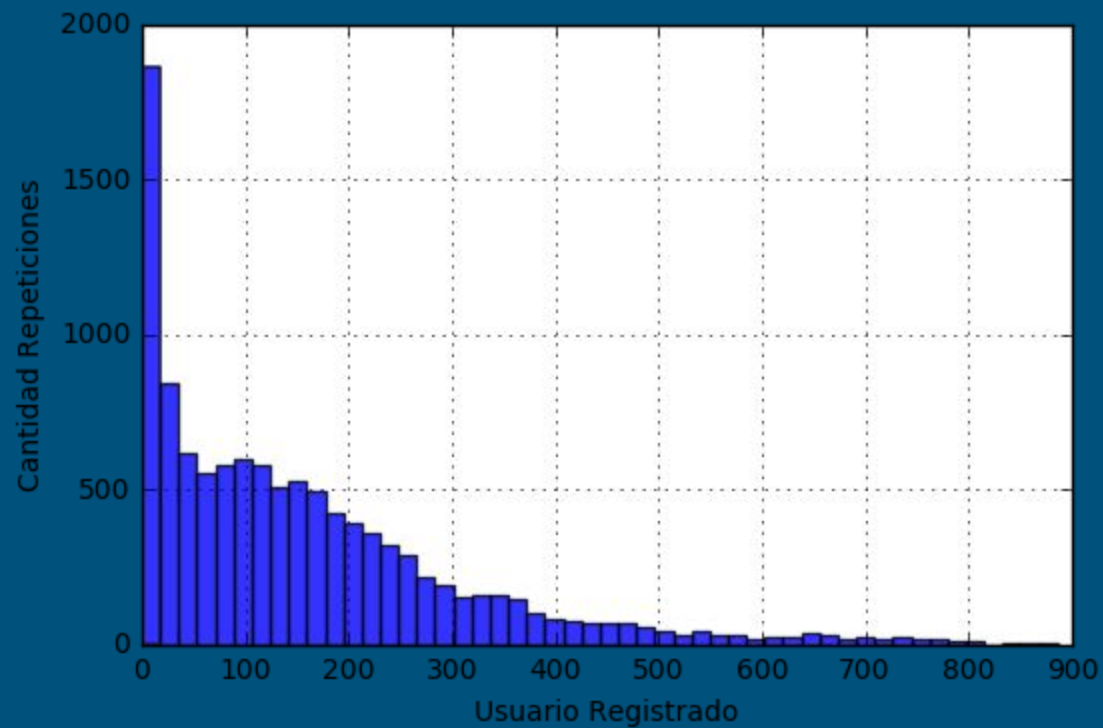


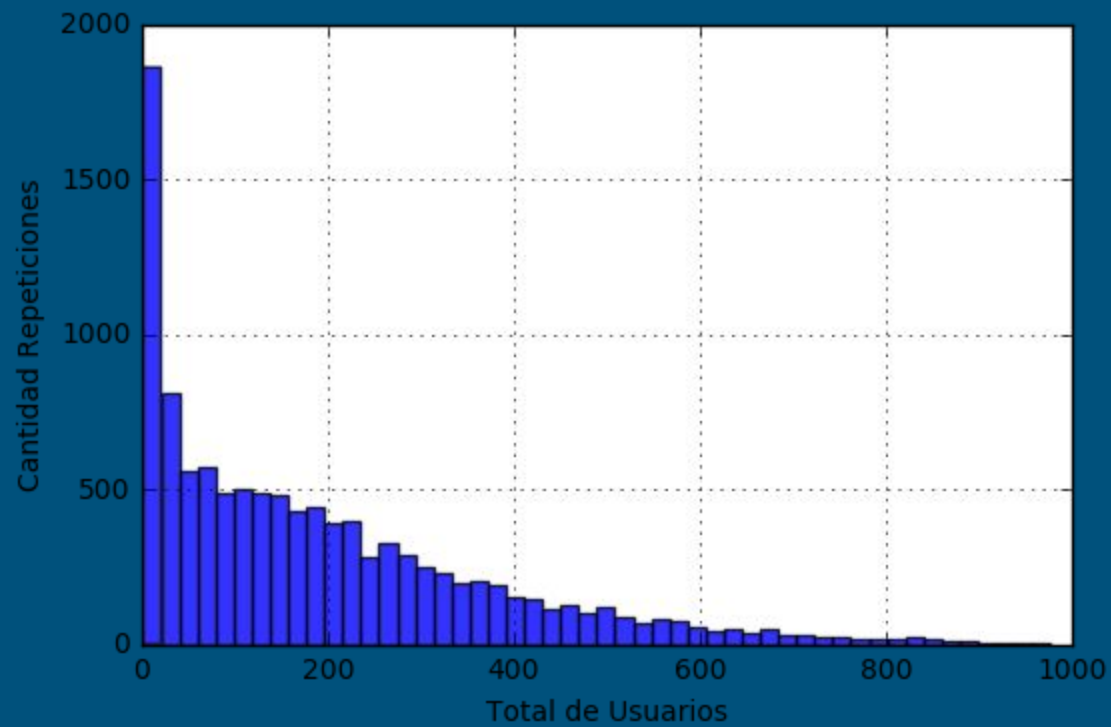






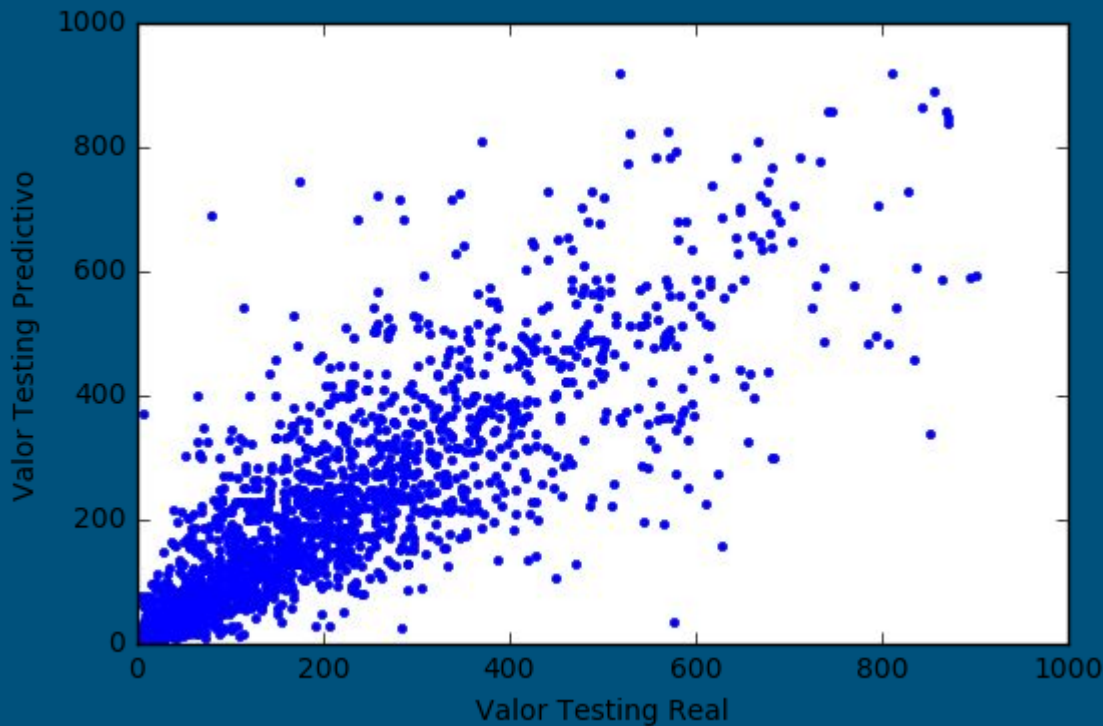




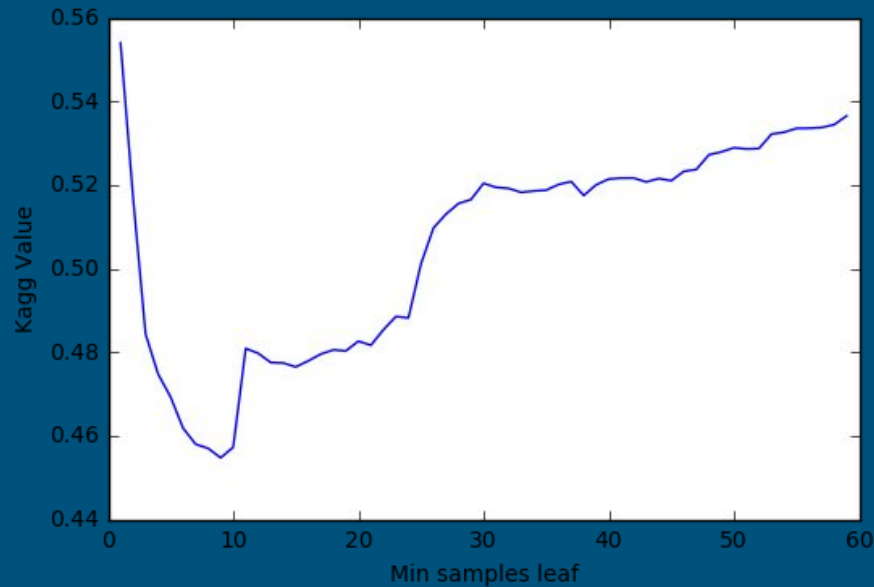
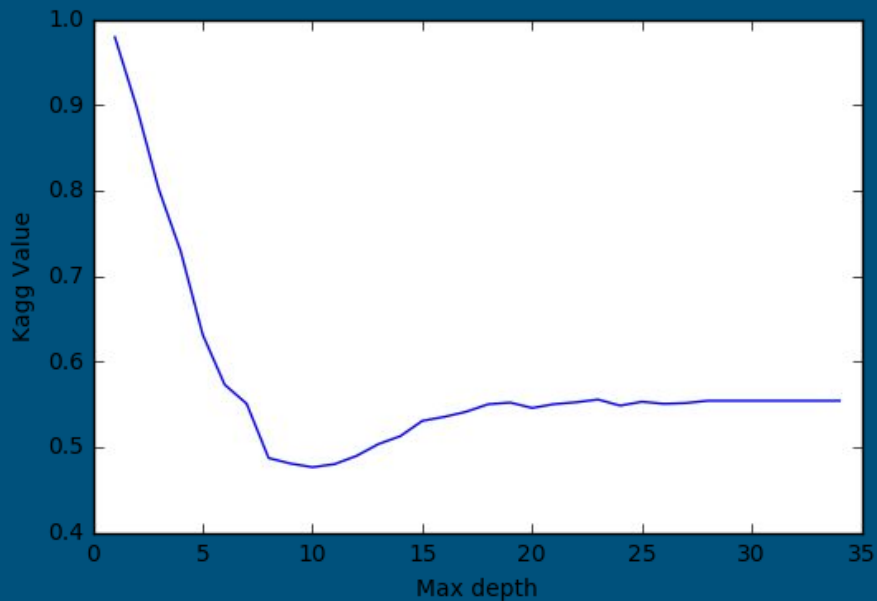


# Entrenar un árbol de regresión

```
SCORE TEST=0.703261  
KAGG EVAL TRAIN =0.028516  
KAGG EVAL TEST =0.574539  
KAGG EVAL VALIDATION =0.553973
```



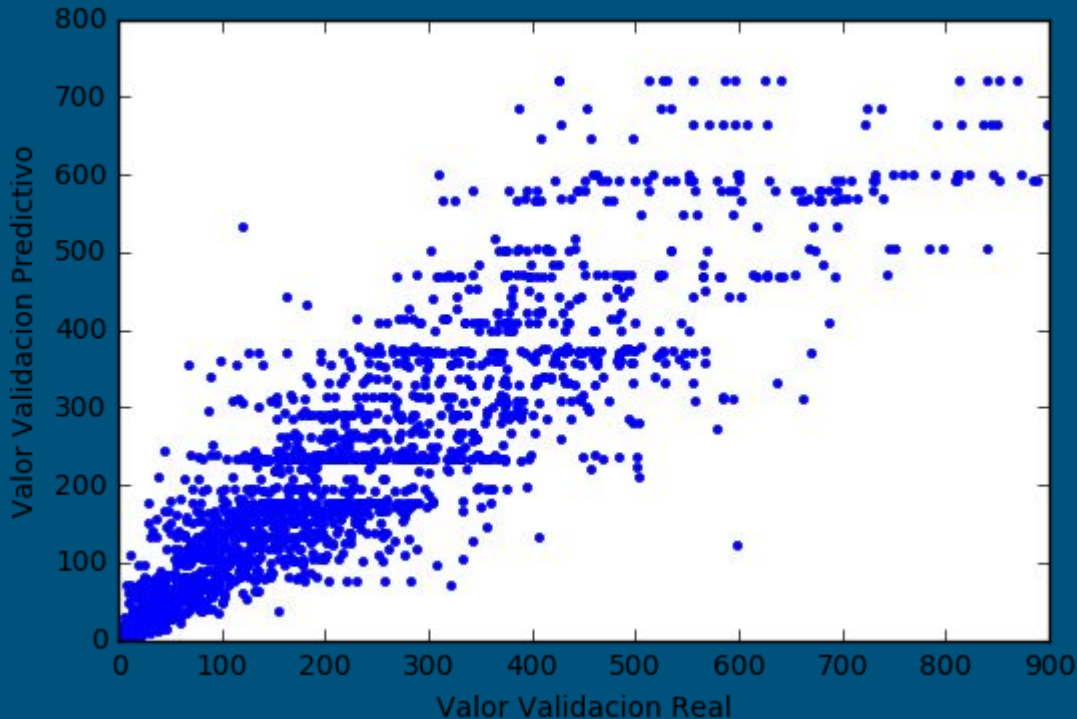
# Mejorar el árbol (hiper-parámetros)



# Mejorar el árbol (hiper-parámetros)

Mejor valor para máximo niveles: 10  
Mejor valor para mínimo de datos por hoja: 9

KAGG EVAL VAL =0.451062



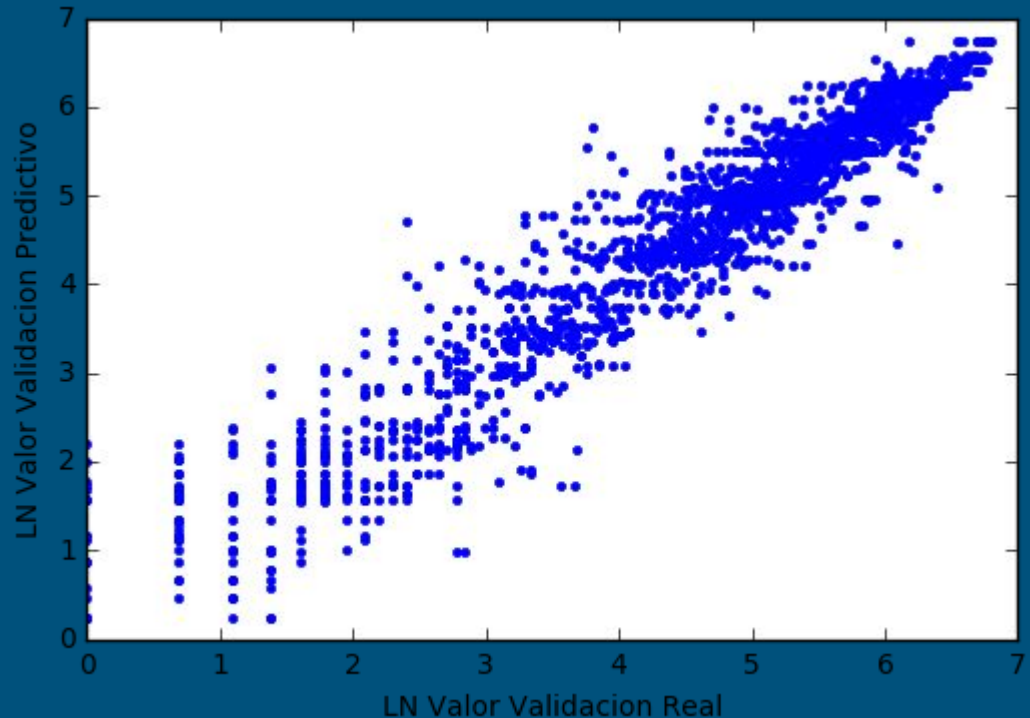
# Mejorar el árbol de regresión (datos)

---

- Se extrajo mayor información desde los datos: mes y año.
- Se aplicó la función logaritmo natural a los datos de conteo de préstamos de bicicletas (count).
- Se intentó además aplicar funciones de logaritmo o raíz a la velocidad del viento y ajustar a una normal los valores de las temperaturas, pero no se obtuvo tan buenos resultados por lo que se excluyeron.

# Mejorar el árbol de regresión (datos)

KAGG EVAL VALIDATION =0.402686





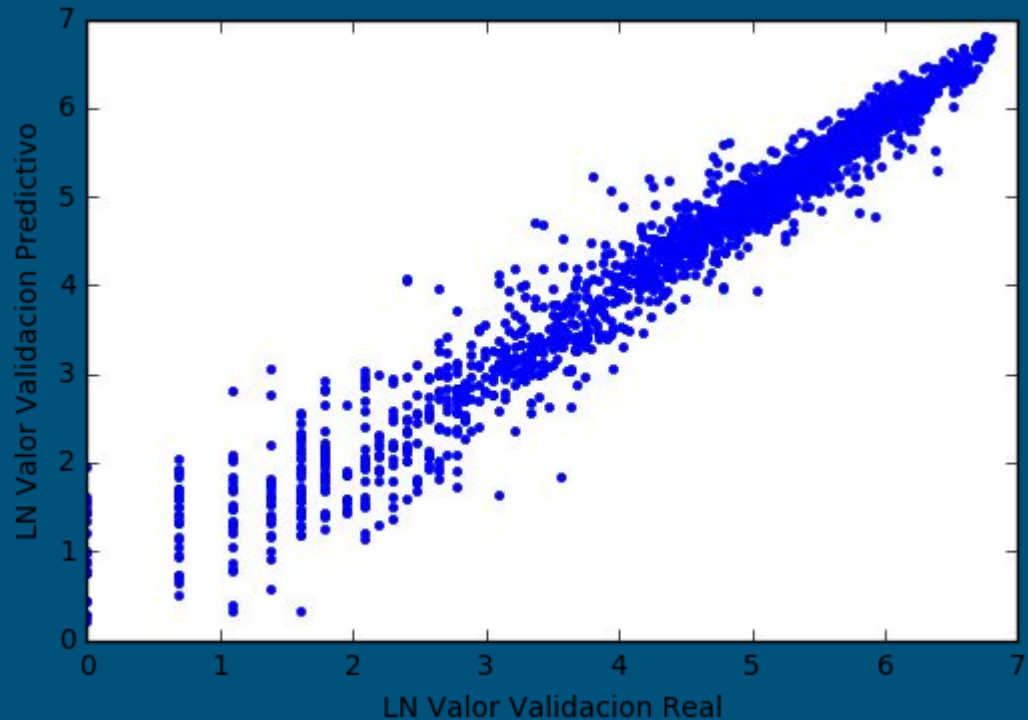
# Entrenar SVM no lineal

---

- Se entrenó con parámetros por defecto ( $C=1$ ,  $\epsilon=0.1$ ).
- Los datos anteriormente manipulados recibieron un escalamiento para hacerlos comparables.
- Los datos categóricos recibieron una transformación de tal forma que queden representados como valores binarios, trinaros u otros dependiendo de la cantidad de valores posibles a tomar.

# Entrenar SVM no lineal

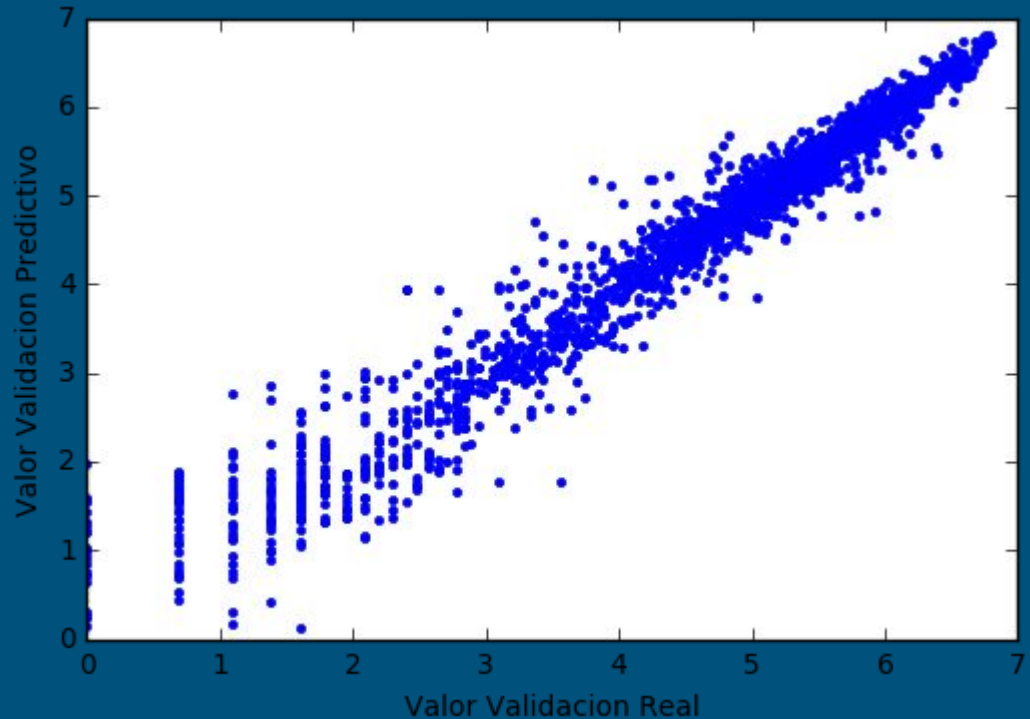
KAGG EVAL VALIDATION =0.292625



# Mejorar la SVM (parámetros)

Mejores valores:  $C = 1.600000$  ,  $\epsilon = 0.050000$ ,

KAGG EVAL TRAIN =0.220828  
KAGG EVAL VAL =0.287996



# Evaluar usando Cross Validation(árbol)

---

```
-----TREE-----  
MAX depth= 9    MIN leaf= 3    => kaggeval= 0.452973  
MAX depth= 10   MIN leaf= 3    => kaggeval= 0.437493  
MAX depth= 11   MIN leaf= 3    => kaggeval= 0.432979  
MAX depth= 12   MIN leaf= 6    => kaggeval= 0.430801  
MAX depth= 13   MIN leaf= 3    => kaggeval= 0.429883  
MAX depth= 13   MIN leaf= 6    => kaggeval= 0.427722  
MAX depth= 14   MIN leaf= 6    => kaggeval= 0.427608  
MAX depth= 15   MIN leaf= 6    => kaggeval= 0.427535  
MAX depth= 16   MIN leaf= 6    => kaggeval= 0.427143
```

# Evaluar usando Cross Validation(SVM)

---

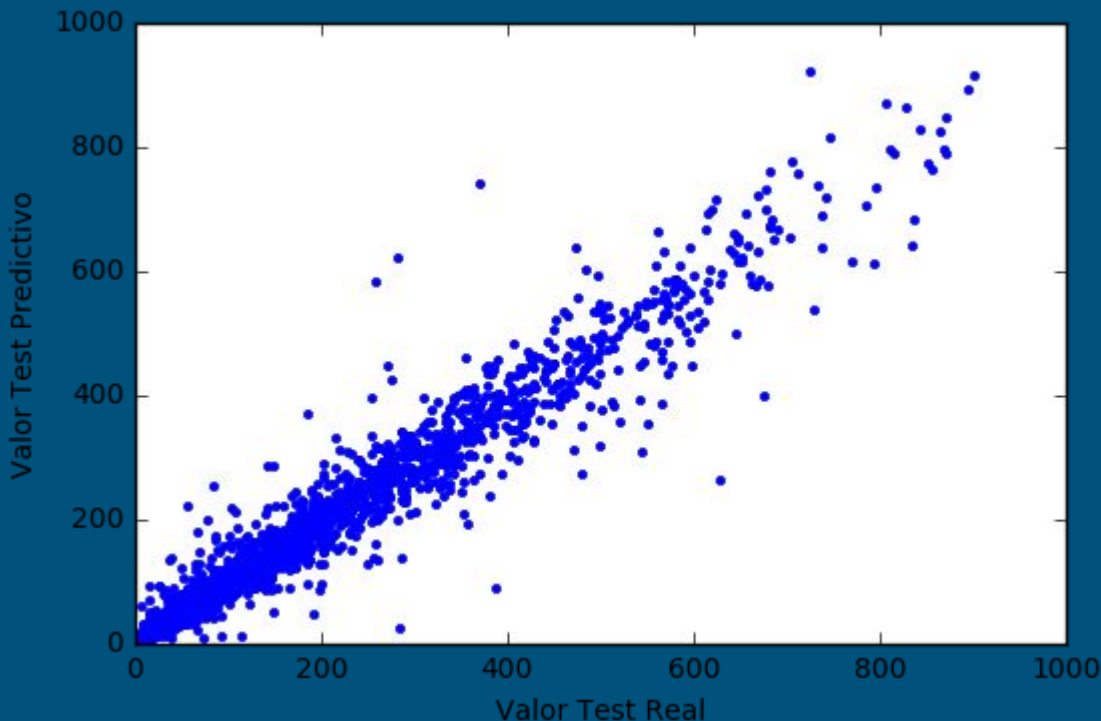
```
-----SVM-----  
C= 1.000000  epsilon = 0.050000  =>  kaggeval= 0.305019  
C= 1.200000  epsilon = 0.050000  =>  kaggeval= 0.303537  
C= 1.200000  epsilon = 0.100000  =>  kaggeval= 0.303391  
C= 1.400000  epsilon = 0.050000  =>  kaggeval= 0.302621  
C= 1.400000  epsilon = 0.100000  =>  kaggeval= 0.302329  
C= 1.500000  epsilon = 0.100000  =>  kaggeval= 0.302113  
C= 1.600000  epsilon = 0.100000  =>  kaggeval= 0.302070
```

# Ensamblar dos máquinas según predicción de tipo de usuario

- Entrenando dos máquinas SVM especializadas en el conteo de personas registradas y otra que no (casuales).
- Esto mismo podría aplicarse con otros tipos de especialización, como considerar todo lo que tenga que referencia con fechas y horas, o con lo que tiene que ver con clima y dificultades físicas y prácticas para los ciclistas.

# Ensamblar dos máquinas según predicción de tipo de usuario

KAGG EVAL TEST =0.307498



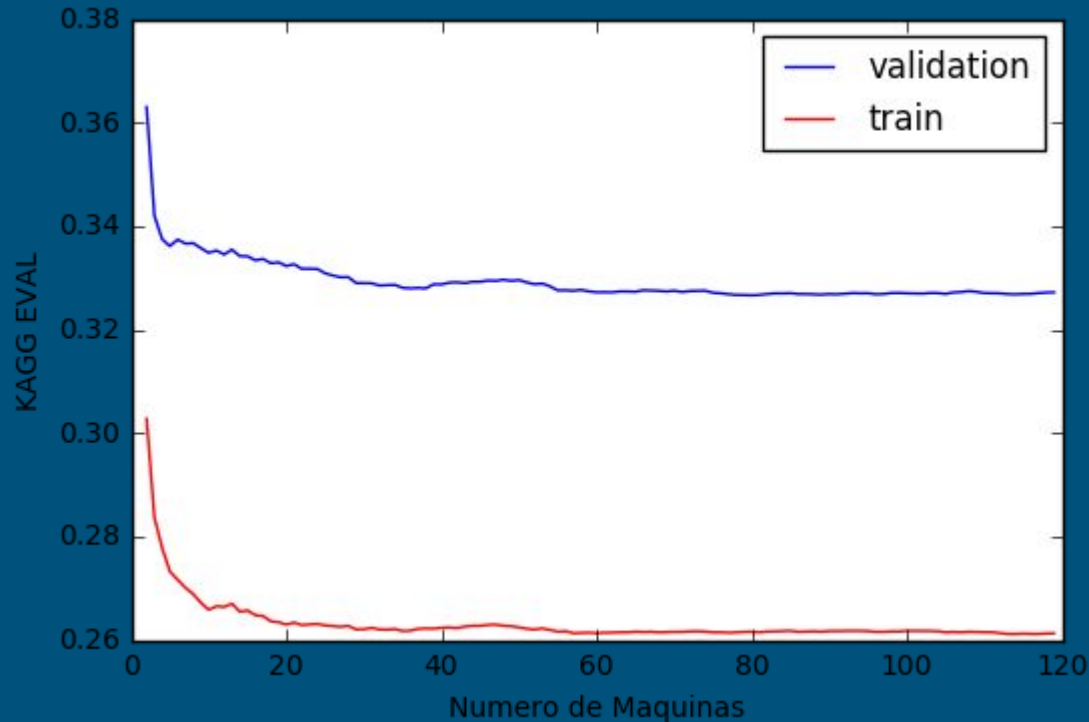
# Evaluar algoritmo genérico de ensamblado

Para 5 Máquinas se obtiene KAGG EVAL TRAIN = 0.273225  
Para 6 Máquinas se obtiene KAGG EVAL TRAIN = 0.271616  
Para 7 Máquinas se obtiene KAGG EVAL TRAIN = 0.270044  
Para 8 Máquinas se obtiene KAGG EVAL TRAIN = 0.268881  
Para 9 Máquinas se obtiene KAGG EVAL VAL = 0.335720  
Para 9 Máquinas se obtiene KAGG EVAL TRAIN = 0.267157  
Para 10 Máquinas se obtiene KAGG EVAL VAL = 0.334814  
Para 10 Máquinas se obtiene KAGG EVAL TRAIN = 0.265822  
Para 12 Máquinas se obtiene KAGG EVAL VAL = 0.334478  
Para 14 Máquinas se obtiene KAGG EVAL VAL = 0.334191  
Para 14 Máquinas se obtiene KAGG EVAL TRAIN = 0.265456  
Para 15 Máquinas se obtiene KAGG EVAL VAL = 0.334136  
Para 16 Máquinas se obtiene KAGG EVAL VAL = 0.333387  
Para 16 Máquinas se obtiene KAGG EVAL TRAIN = 0.264784  
Para 17 Máquinas se obtiene KAGG EVAL TRAIN = 0.264611  
Para 18 Máquinas se obtiene KAGG EVAL VAL = 0.332804  
Para 18 Máquinas se obtiene KAGG EVAL TRAIN = 0.263571  
Para 19 Máquinas se obtiene KAGG EVAL TRAIN = 0.263400  
Para 20 Máquinas se obtiene KAGG EVAL VAL = 0.332268  
Para 20 Máquinas se obtiene KAGG EVAL TRAIN = 0.262958

Para 25 Máquinas se obtiene KAGG EVAL VAL = 0.330871  
Para 25 Máquinas se obtiene KAGG EVAL TRAIN = 0.262782  
Para 26 Máquinas se obtiene KAGG EVAL VAL = 0.330431  
Para 26 Máquinas se obtiene KAGG EVAL TRAIN = 0.262662  
Para 27 Máquinas se obtiene KAGG EVAL VAL = 0.330087  
Para 27 Máquinas se obtiene KAGG EVAL TRAIN = 0.262518  
Para 29 Máquinas se obtiene KAGG EVAL VAL = 0.328972  
Para 29 Máquinas se obtiene KAGG EVAL TRAIN = 0.262024  
Para 30 Máquinas se obtiene KAGG EVAL VAL = 0.328946  
Para 31 Máquinas se obtiene KAGG EVAL VAL = 0.328942  
Para 32 Máquinas se obtiene KAGG EVAL VAL = 0.328493  
Para 32 Máquinas se obtiene KAGG EVAL TRAIN = 0.262004  
Para 33 Máquinas se obtiene KAGG EVAL TRAIN = 0.261951  
Para 35 Máquinas se obtiene KAGG EVAL VAL = 0.328041  
Para 35 Máquinas se obtiene KAGG EVAL TRAIN = 0.261687  
Para 36 Máquinas se obtiene KAGG EVAL VAL = 0.327956  
Para 38 Máquinas se obtiene KAGG EVAL VAL = 0.327935  
Para 55 Máquinas se obtiene KAGG EVAL VAL = 0.327548  
Para 55 Máquinas se obtiene KAGG EVAL TRAIN = 0.261574



# Evaluar algoritmo genérico de ensamblado



---

### 3. Reconocimiento de Imágenes en CIFAR10

# Contexto

---

El dataset CIFAR10 está compuesto por 60.000 imágenes RGB de 32x32 píxeles el cual contiene 10 clases de objetos (6.000 ejemplos por clase). Las clases corresponden a: Gato, Perro, Rana, Caballo, Pájaro, Ciervo, Avión, Automóvil, Camión y Barco.

El conjunto de datos de entrenamiento corresponde a 50.000 imágenes, el de prueba a 10.000 imágenes y el de validación a 10.000 imágenes.

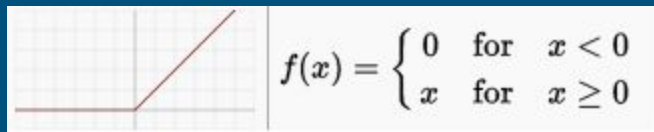
# Carga de datos y escalamiento

---

Se carga los datos de CIFAR10 en 2 matrices de entrenamiento ( $50.000 \times 32 \times 32$ ), 2 matrices de prueba ( $10.000 \times 32 \times 32$ ) y 2 matrices de validación ( $10.000 \times 32 \times 32$ ). Luego se escalan estas matrices dividiendo los valores RGB en 255.

# Red Neuronal para clasificación de CIFAR10

Se entrena una red neuronal con 1 capa oculta de 100 neuronas el cual utiliza una función de activación rectificadora (ReLU):



El resultado de esta red neuronal fue el siguiente:

```
9952/10000 [=====>.] - ETA: 0s  
Accuracy de train: 0.819920  
Accuracy de validacion: 0.820320  
Accuracy de test: 0.820000
```

# Red neuronal con Histograma de Color

---

```
40000/40000 [=====] - 1s  
 9888/10000 [=====>.] - ETA: 0s  
Accuracy de train: 0.900000  
Accuracy de validacion: 0.900000  
Accuracy de test: 0.900000
```

# Red neuronal con HOG

---

```
9824/10000 [=====>.] - ETA: 0s  
Accuracy de train: 0.939097  
Accuracy de validacion: 0.923510  
Accuracy de test: 0.923050
```

# Red neuronal con Histograma de Color y HOG

---

```
9536/10000 [======>...] - ETA: 0s  
Accuracy de train: 0.941872  
Accuracy de validacion: 0.926120  
Accuracy de test: 0.924690
```



# SVM no lineal y árbol de clasificación

---

