



Módulo 5 – Aprendizaje de Máquina Supervisado

Regresión Logística

Especialización en Ciencia de Datos

Regresión Logística

La regresión logística es un algoritmo de **aprendizaje automático supervisado**, utilizado para resolver problemas de **clasificación binaria**. Es decir, es utilizado para **predecir la probabilidad** de que un ejemplo pertenezca a una de dos clases posibles, donde la variable dependiente (la clase) es binaria (por ej.: verdadero/falso, sí/no, 0/1).

Sí / No
1 / 0
True / False

Clasificación Binaria

33% Sí - 67% No

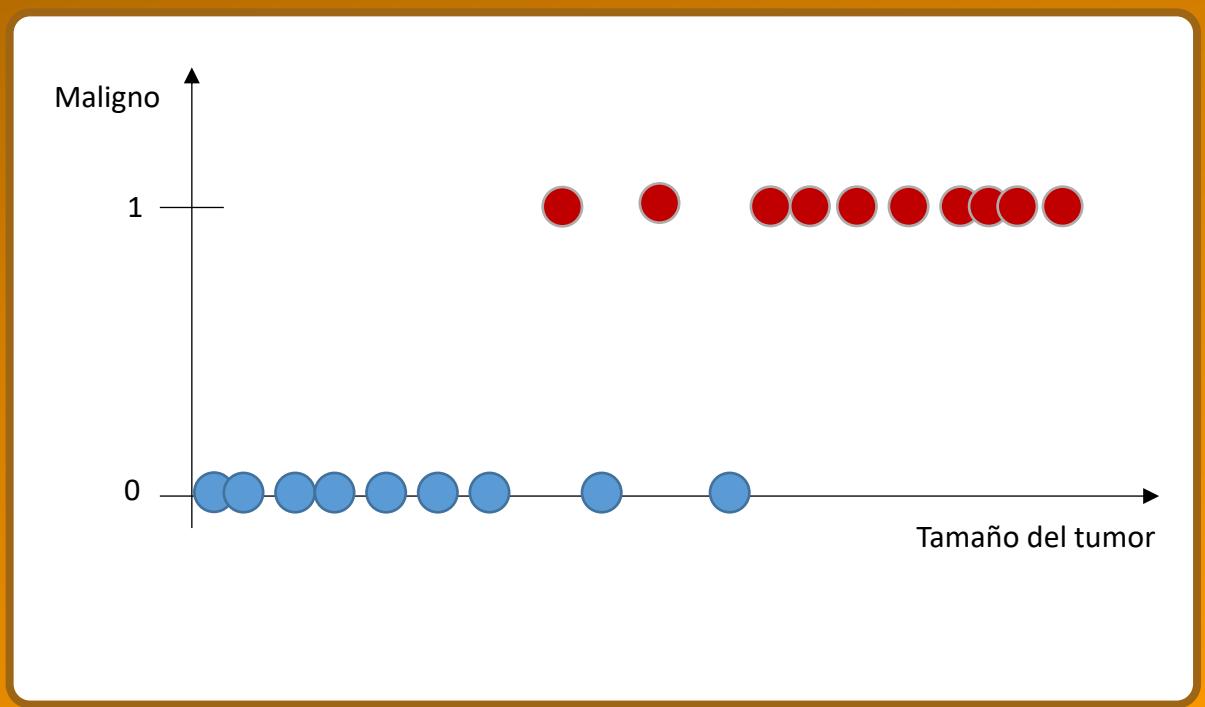
50% 1 - 50% 0

75% True - 25% False

Predecir Probabilidad

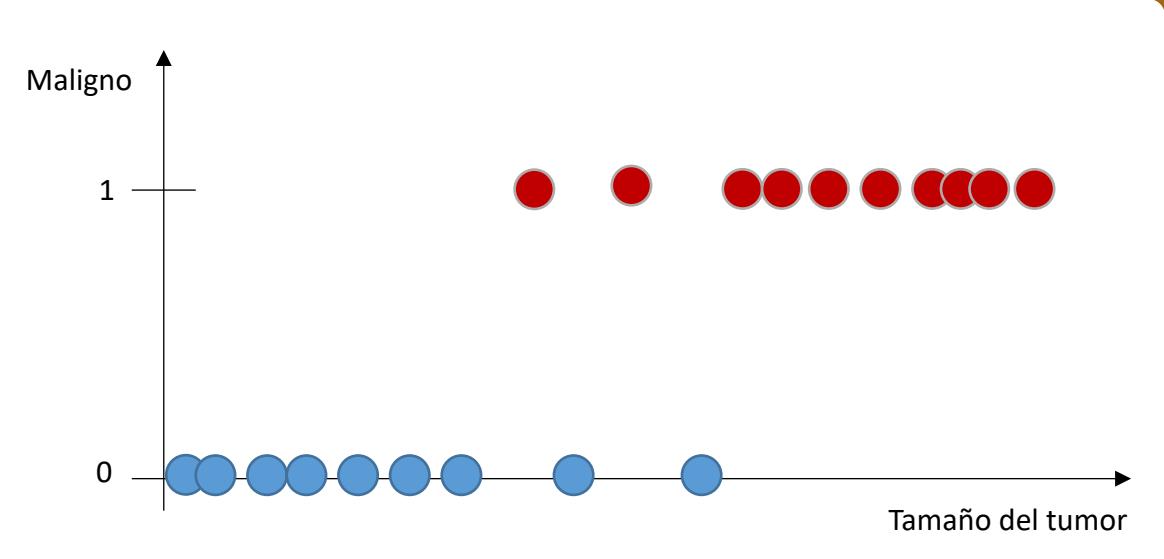
Regresión Logística

- Tomemos el siguiente ejemplo, en el cual queremos modelar la probabilidad de una mujer, que se le ha detectado un tumor en los pechos, que éste sea maligno o benigno. Para esto, se han tomado una serie de mediciones y se han graficado en el siguiente diagrama.
- Como se puede observar, cuando los tumores tienen un tamaño pequeño, es más probable que sea benigno, pero cuando su tamaño es más grande, es aumenta la probabilidad que sea maligno.



Regresión Logística

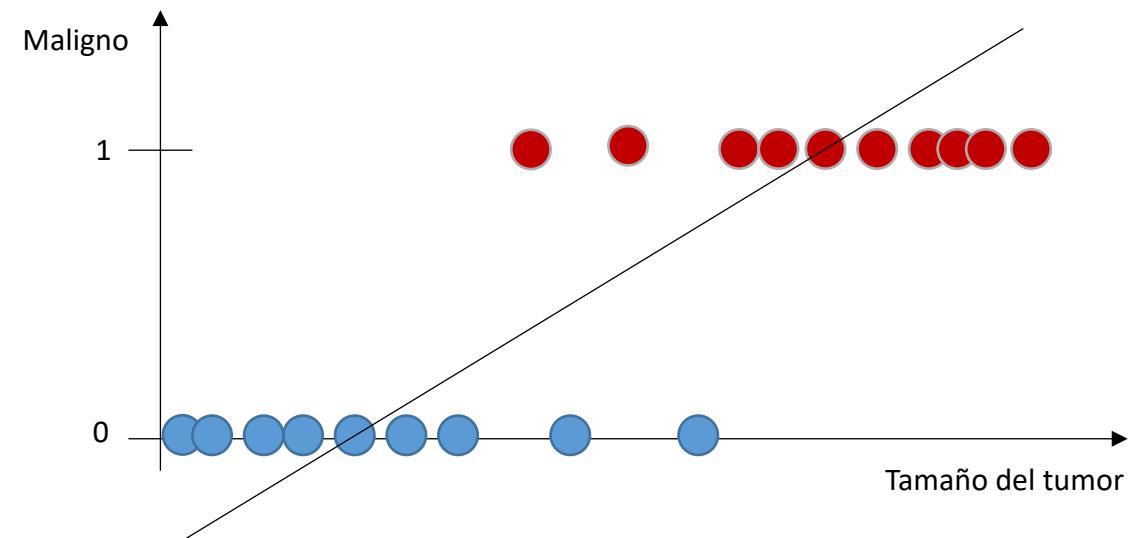
- Entonces, el objetivo es elaborar un modelo probabilístico que permita obtener la probabilidad de que sea maligno a partir del tamaño del tumor.



Regresión Logística

Una opción sería trazar una línea recta y definir una función de probabilidad con los parámetros b_0 y b_1 .

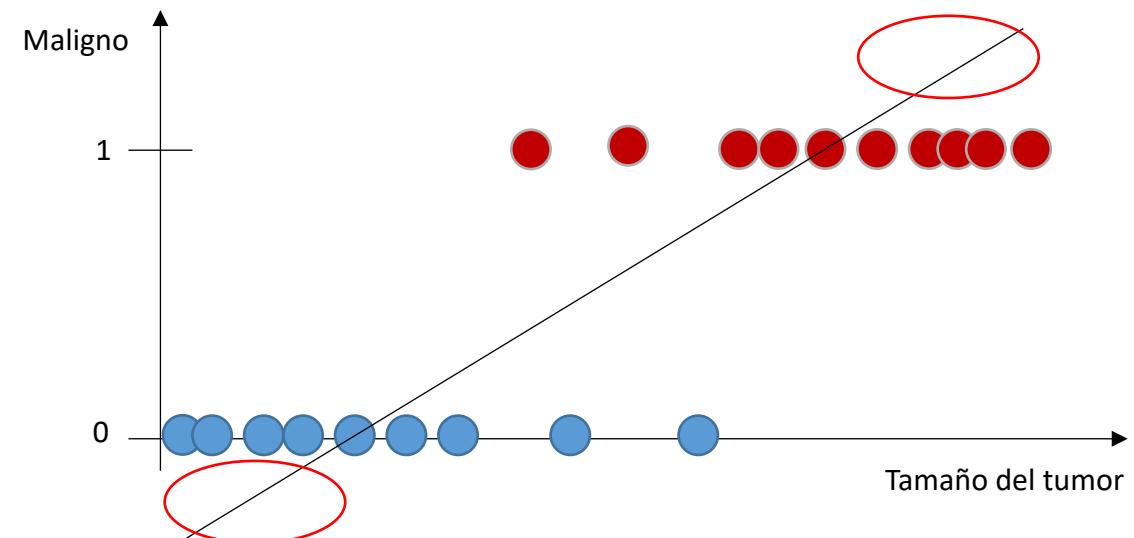
$$P(y) = b_0 + b_1 x_1$$



Regresión Logística

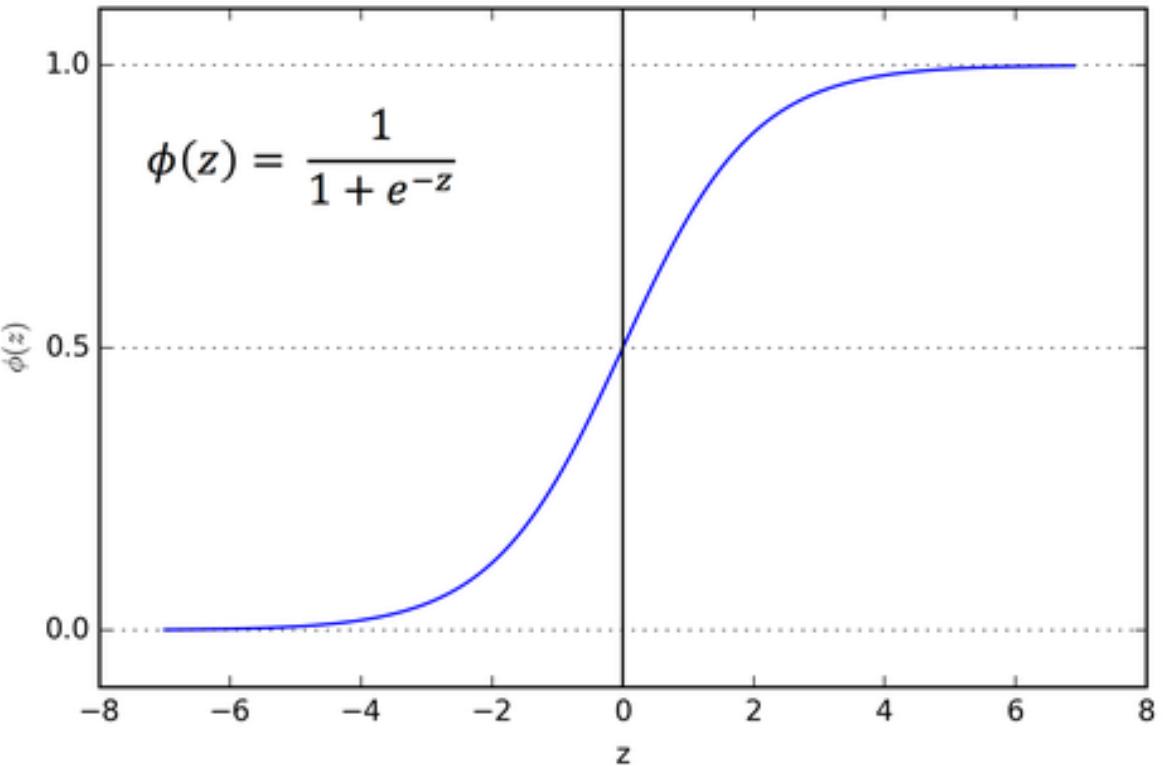
No obstante, utilizar una línea recta provocaría que, dependiendo de los parámetros, podríamos obtener valores de probabilidad menor que cero y mayor que uno, en algunos casos.

$$P(y) = b_0 + b_1 x_1$$

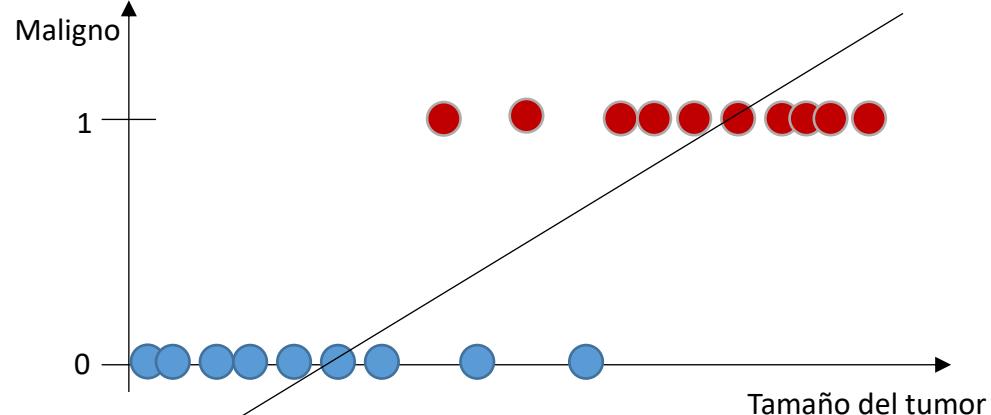


Función Sigmoidea

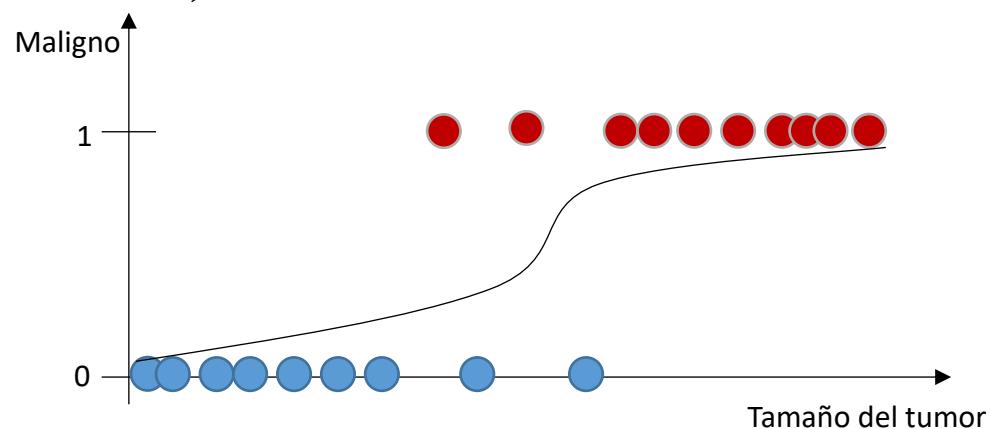
Para sortear la situación anterior, sería más apropiado utilizar la función Sigmoidea (o Logística), pues posee una característica que puede ayudarnos a resolver nuestro problema. Esto es, que para todo valor de X, su resultado se encuentra entre 0 y 1. Cuando z tiende a infinito, la probabilidad tiende 1, y viceversa, cuando z tiende a menos infinito, la probabilidad tiende a 0.



Aplicando Función Sigmoidea



Modelo Lineal
 $y = b_0 + b_1 x_1$



Modelo Logístico

$$p = \frac{1}{1 + e^{-(b_0 + b_1 x_1)}}$$

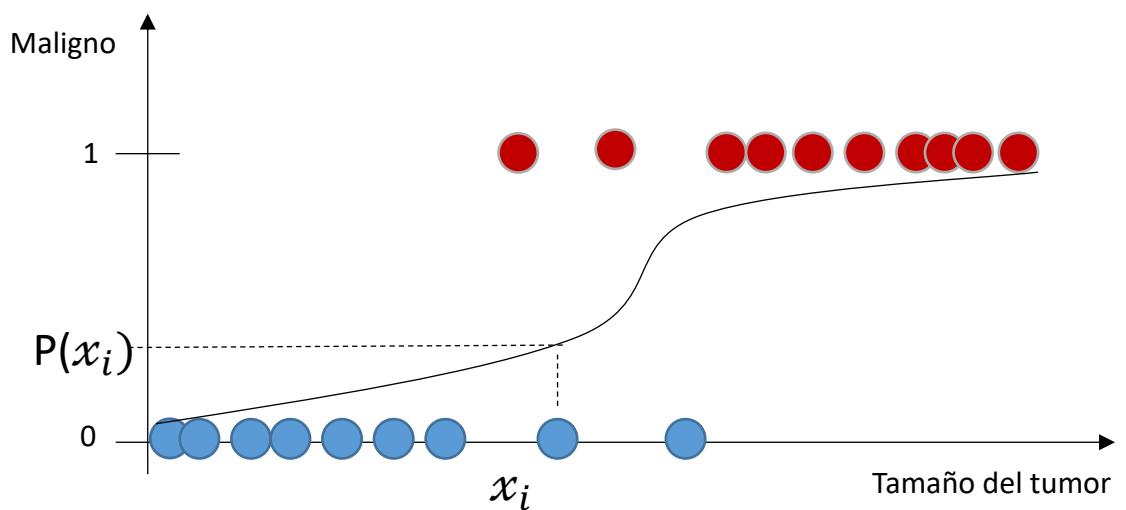
Si aplicamos la función sigmoidea a "y", obtenemos el modelo logístico.

Aplicando Función Sigmoidea

- Con el modelo logístico resolvemos el problema, y para cada valor de x obtendremos una probabilidad $p(x)$ que se encuentra entre 0 y 1.

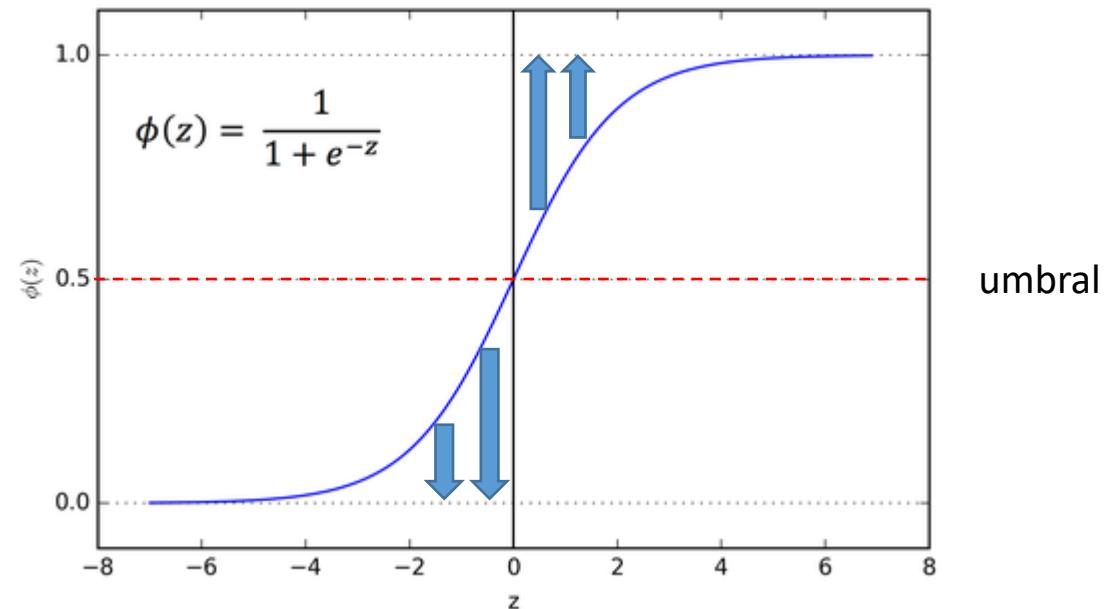
Modelo Logístico

$$p(x) = \frac{1}{1 + e^{-(b_0 + b_1 x)}}$$



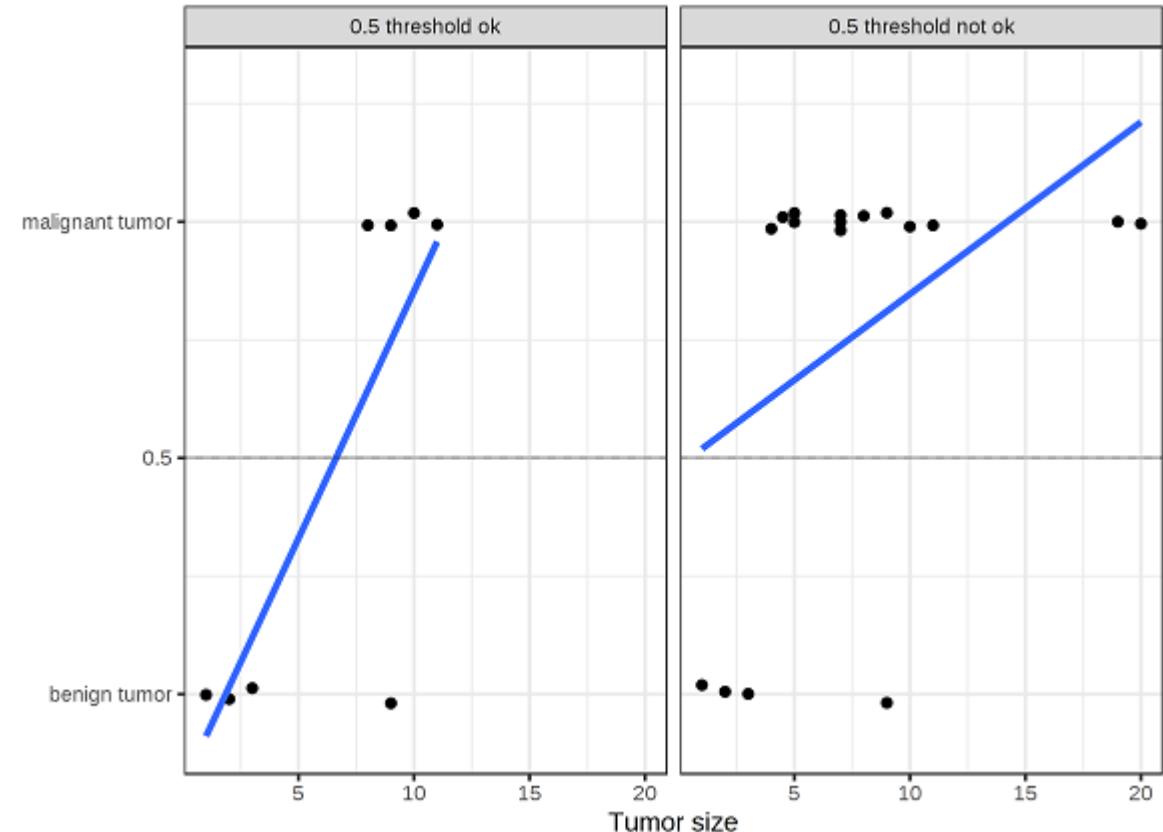
Modelo de Clasificación

Para elaborar un modelo de clasificación, que retorne valores discretos 0 o 1, simplemente definimos un punto de corte en 0.5, es decir. Todo valor sobre 0.5 se aproxima a 1, y todo valor bajo 0.5 se aproxima a 0.



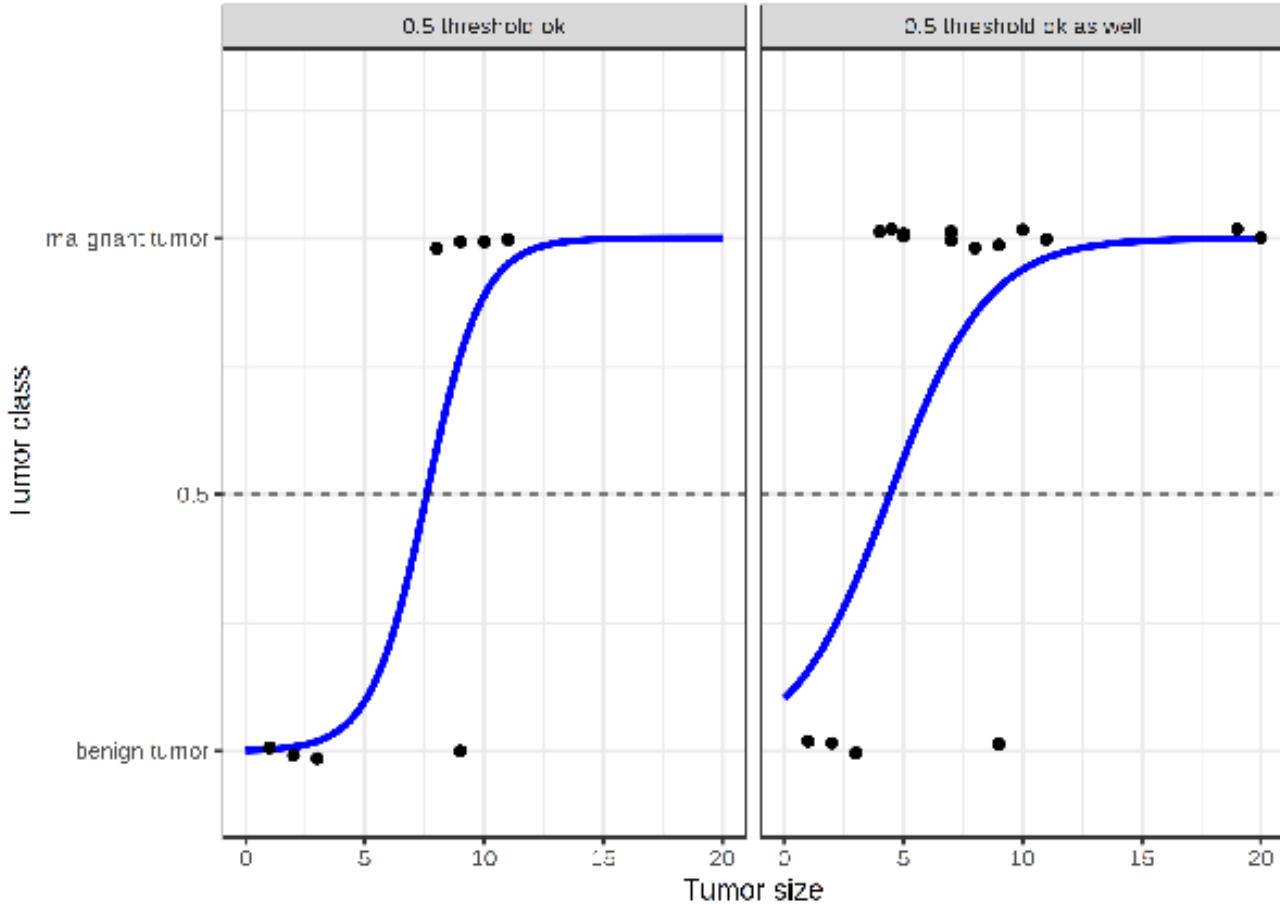
Regresión Logística

Otra ventaja de utilizar la función sigmoidea en vez de una recta, es que las rectas son más sensibles a los valores de puntos outliers, tal como se aprecia en la siguiente figura, lo cual puede hacer cambiar drásticamente al modelo.



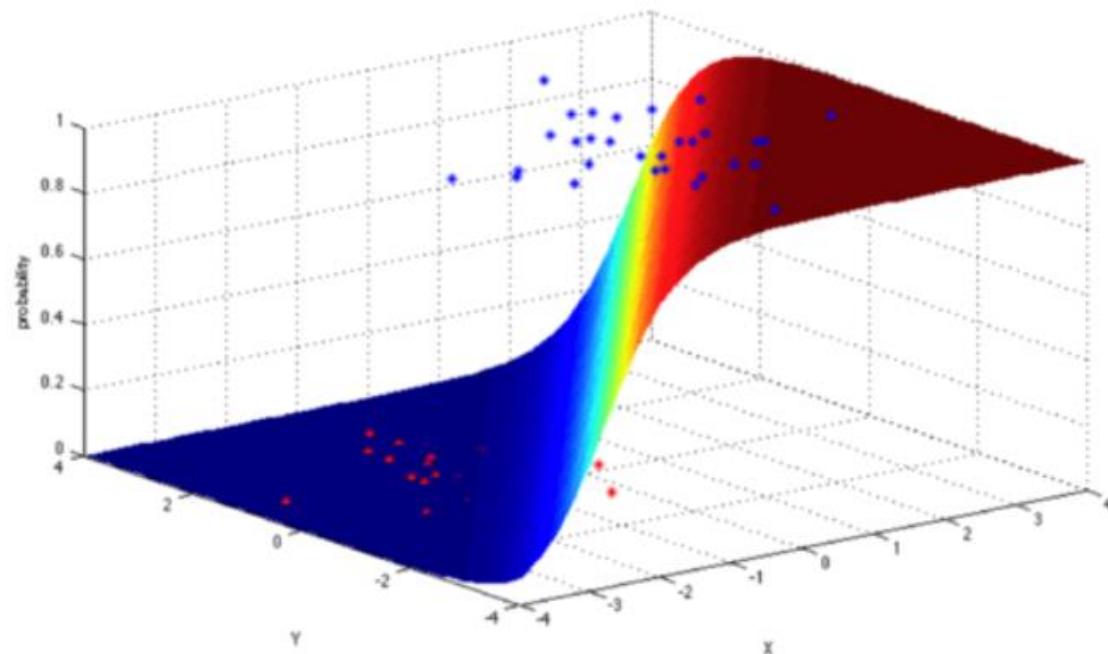
Regresión Logística

En esta figura se aprecia que el efecto de los outliers afecta menos al modelo, lo cual lo hace más generalizable, al no ser tan dependiente de los datos más alejados.



Regresión Logística

Por último, en el siguiente diagrama se puede apreciar cómo la regresión logística realiza una separación de las clases en un espacio con dos dimensiones.



Ventajas y Desventajas

1. Interpretabilidad: La regresión logística es fácil de interpretar, ya que los coeficientes se pueden utilizar para identificar la importancia relativa de cada variable predictora en la predicción de la clase objetivo.

2. Eficiencia: La regresión logística es computacionalmente eficiente y puede manejar grandes conjuntos de datos con rapidez.

3. Flexibilidad: La regresión logística puede ser extendida a modelos más complejos para acomodar una variedad de problemas de clasificación, como la clasificación multi-clase.

4. Regularización: La regresión logística se puede regularizar para evitar el sobreajuste y mejorar la generalización del modelo.

1. Linealidad: La regresión logística es un modelo lineal, lo que significa que asume que la relación entre las variables predictoras y la variable objetivo es lineal. Si la relación es no lineal, el modelo puede ser inadecuado.

2. Sensibilidad a valores atípicos: La regresión logística puede ser sensible a los valores atípicos, lo que puede afectar la calidad del modelo.

3. No siempre funciona bien con datos no balanceados: Si los datos de entrenamiento están desequilibrados, es decir, si hay muchas más observaciones en una clase que en la otra, la regresión logística puede tener dificultades para predecir la clase minoritaria.

4. No captura relaciones complejas: La regresión logística no puede capturar relaciones complejas entre las variables predictoras y la variable objetivo, como las interacciones no lineales entre las variables predictoras. Para estos casos se pueden utilizar modelos más avanzados como redes neuronales o árboles de decisión.

Implementación en Python

Obtención de Datos

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline

train = pd.read_csv('titanic.csv')

train.head()
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th... Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S

En este ejemplo, utilizaremos la información de los sobrevivientes del Titanic, con el propósito de implementar un modelo de clasificación que permita predecir si un pasajero, de acuerdo a sus características, se salvaría o no al abordar el Titanic.

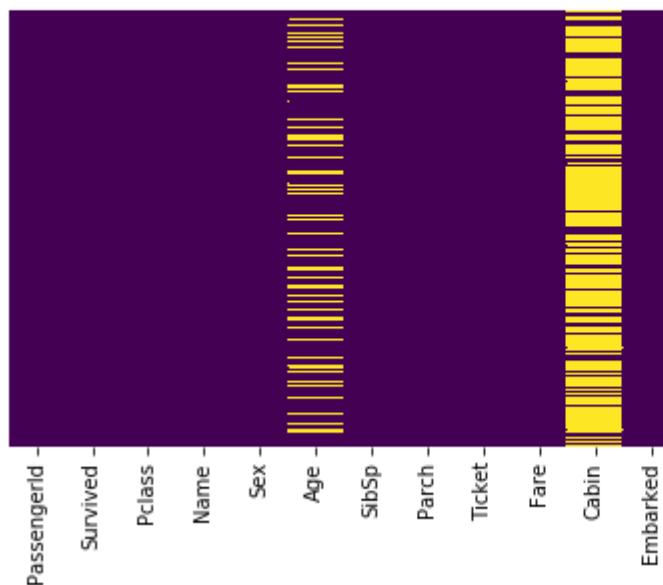
Datos Perdidos

Inspeccionamos rápidamente los datos para ver si existen datos perdidos. A continuación, se muestra una forma rápida de verificar visualmente los datos perdidos.

```
train.isnull().sum()
```

```
PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
Age            177
SibSp            0
Parch            0
Ticket           0
Fare             0
Cabin          687
Embarked        2
dtype: int64
```

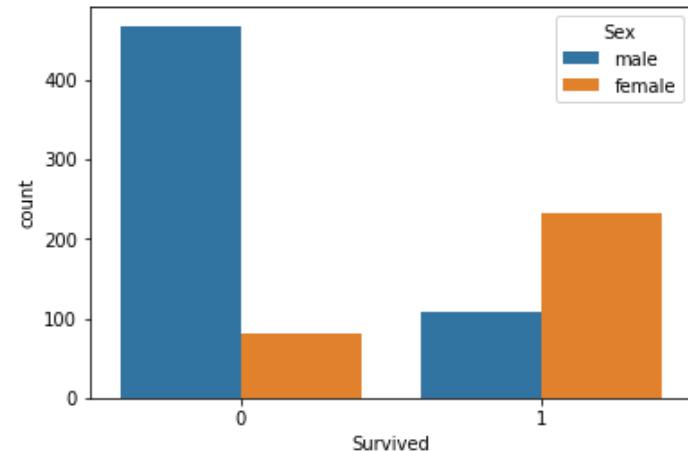
```
sns.heatmap(train.isnull(), yticklabels=False, cbar=False, cmap="viridis")
```



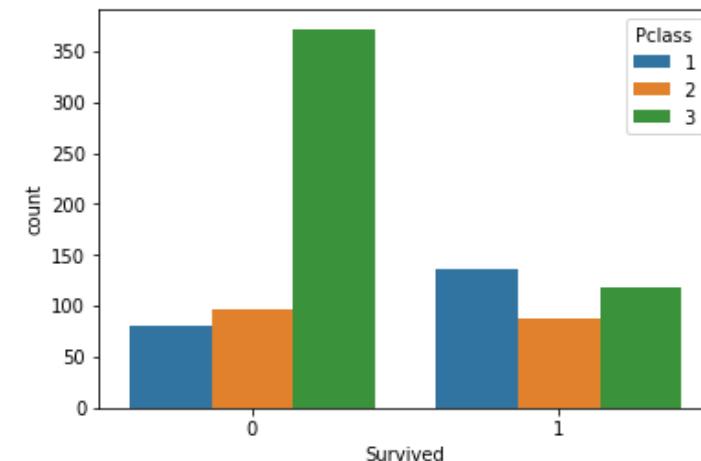
Análisis Exploratorio

Analizamos los sobrevivientes por sexo y por clase de cabina.

```
sns.countplot(x="Survived", hue="Sex", data=train)  
<matplotlib.axes._subplots.AxesSubplot at 0x1f4bd1d7e10>
```



```
sns.countplot(x="Survived", hue="Pclass", data=train)  
<matplotlib.axes._subplots.AxesSubplot at 0x1f4bd3210b8>
```



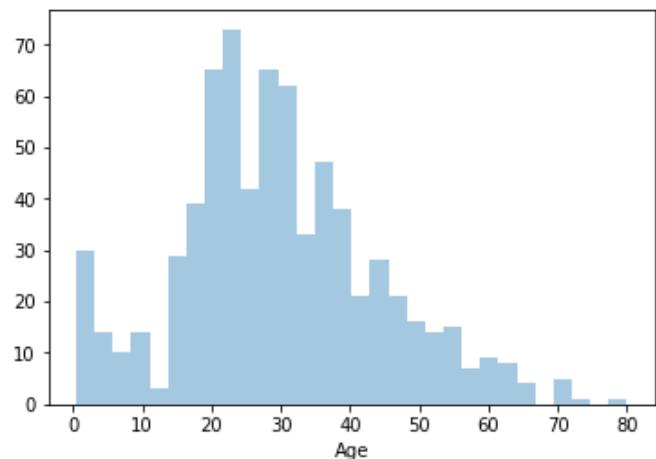
Nótese que sobrevivió una proporción mayor de mujeres que de hombres. Nótese también, que sobrevivió una proporción mayor de personas que viajaban en primera clase.

Análisis Exploratorio

Analizamos la distribución de edad de los pasajeros y la tarifa pagada.

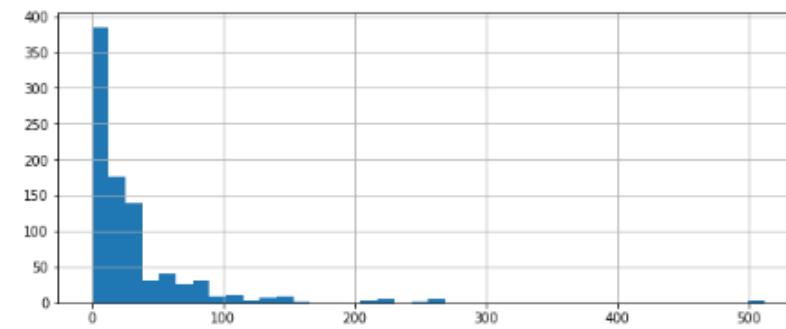
```
sns.distplot(train['Age'].dropna(), bins=30, kde=False)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1f4bd4698d0>
```



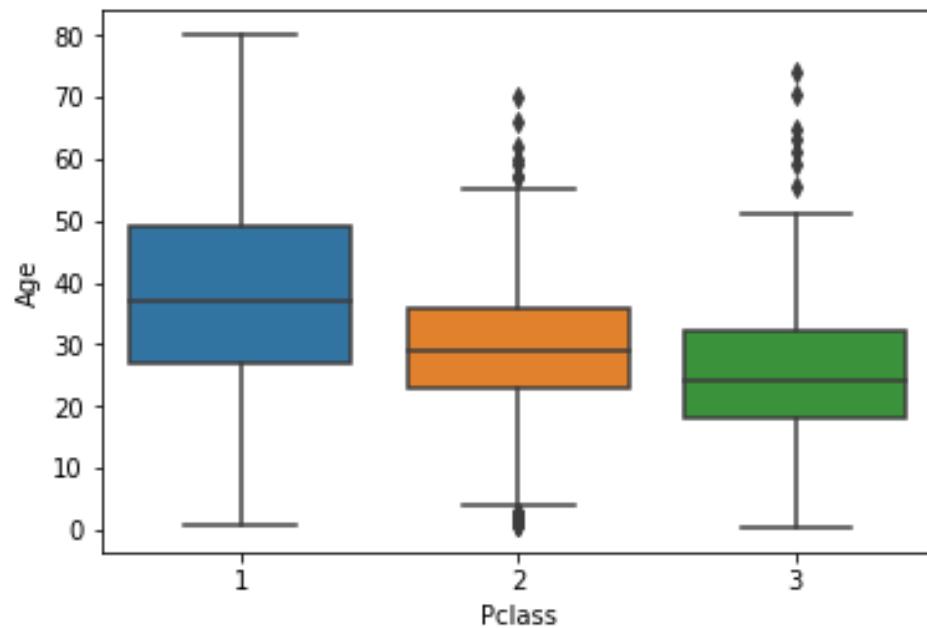
```
train['Fare'].hist(bins=40, figsize=(10,4))
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1f4bd6a5780>
```



Limpiando los Datos

```
sns.boxplot(x='Pclass', y='Age', data=train)  
<matplotlib.axes._subplots.AxesSubplot at 0x1f4be7650b8>
```



Antes de comenzar con la limpieza de datos, analizaremos el promedio de edad agrupado por clase de cabina. Como se puede observar, el promedio de edad de los que viajaron en primera clase era aproximadamente de 37 años, los que viajaron en segunda clase su promedio es de aproximadamente 29 años, y los que viajaron en tercera clase de 24 años.

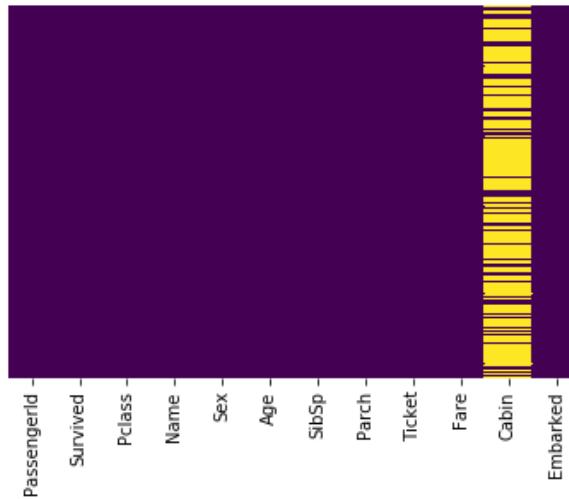
Imputando la Edad

Crearemos una función que pueda asignar la edad de acuerdo a la clase de cabina para aplicar a los valores vacíos del dataset.

```
def imputar_edad(cols):
    edad = cols[0]
    pclass = cols[1]
    if(pd.isnull(edad)):
        if(pclass==1):
            return 37
        if(pclass==2):
            return 29
        if(pclass==3):
            return 24
    else:
        return edad
```

```
train['Age'] = train[['Age', 'Pclass']].apply(imputar_edad, axis=1)

sns.heatmap(train.isnull(), yticklabels=False, cbar=False, cmap="viridis")
<matplotlib.axes._subplots.AxesSubplot at 0x1a90874ae80>
```



Limpando la Matriz

Ahora dejaremos solamente las columnas que nos permitirán hacer una buena predicción. De esta forma, desecharemos la columna Nombre, Ticket, Cabin y PassengerId.

```
train.drop(['Name', 'Ticket', 'Cabin', 'PassengerId'], axis=1, inplace=True)
```

```
train.head()
```

	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
1	1	1	female	38.0	1	0	71.2833	C
3	1	1	female	35.0	1	0	53.1000	S
6	0	1	male	54.0	0	0	51.8625	S
10	1	3	female	4.0	1	1	16.7000	S
11	1	1	female	58.0	0	0	26.5500	S

Binarización de las Variables Categóricas

Probamos cómo quedaría la transformación de las variables categóricas en datos dummy. Recordar que al traspasar una variable categórica a una variable dummy, debemos eliminar una columna para evitar la multicolinealidad en el modelo.

```
pd.get_dummies(train, drop_first=True).head()
```

	Survived	Pclass	Age	SibSp	Parch	Fare	Sex_male	Embarked_Q	Embarked_S
1	1	1	38.0	1	0	71.2833	0	0	0
3	1	1	35.0	1	0	53.1000	0	0	1
6	0	1	54.0	0	0	51.8625	1	0	1
10	1	3	4.0	1	1	16.7000	0	0	1
11	1	1	58.0	0	0	26.5500	0	0	1

Formulación del Modelo

Ahora definimos la matriz de features “X”, y el vector de resultado “y”.

```
X = pd.get_dummies(train, drop_first=True).drop('Survived', axis=1)  
y = train['Survived']
```

```
X.head()
```

Pclass	Age	SibSp	Parch	Fare	Sex_male	Embarked_Q	Embarked_S
1	38.0	1	0	71.2833	0	0	0
3	35.0	1	0	53.1000	0	0	1
6	54.0	0	0	51.8625	1	0	1
10	4.0	1	1	16.7000	0	0	1
11	58.0	0	0	26.5500	0	0	1

Validación Cruzada

Al igual que en los modelos de regresión lineal, dividimos nuestro data set en Training y Test set.

```
from sklearn.model_selection import train_test_split  
  
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=101)
```

Entrenando el Clasificador Logístico

Ahora formulamos nuestro modelo instanciando al clasificador Logístico. Posteriormente, ajustamos nuestro modelo con los datos de entrenamiento y realizamos predicciones en el set de datos de testing.

```
from sklearn.linear_model import LogisticRegression

lm = LogisticRegression()

lm.fit(X_train, y_train)

LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                    intercept_scaling=1, max_iter=100, multi_class='ovr', n_jobs=1,
                    penalty='l2', random_state=None, solver='liblinear', tol=0.0001,
                    verbose=0, warm_start=False)

predictions = lm.predict(X_test)

predictions

array([1, 0, 1, 1, 0, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1,
       0, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 0, 0, 0, 1, 0, 1, 1, 1, 1, 1,
       0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1], dtype=int64)
```

Como se puede observar, las predicciones hechas por el clasificador logístico contienen solamente valores 0 y 1.

Realizando Predicciones de Probabilidad

- Como se puede observar, las predicciones hechas por el clasificador logístico contienen solamente valores 0 y 1. Sin embargo, si queremos hacer una predicción de la probabilidad de sobrevivir, entonces debemos utilizar el método predict_proba().
- En el primer registro, la probabilidad de no sobrevivir (clase 0) es de 88.4%, mientras que la probabilidad de sobrevivir (clase 1) es de 11.5%. En este caso, el clasificador etiquetó al pasajero con clase 0 (No sobrevive).
- En este registro, la probabilidad de no sobrevivir (clase 0) es de 48.9%, mientras que la probabilidad de sobrevivir (clase 1) es de 51.08%. En este caso, el clasificador etiquetó al pasajero con clase 1 (Sobrevive), pero en realidad podríamos decir que tenía una probabilidad casi igual de sobrevivir que de no sobrevivir.

```
# predicciones de probabilidad
predicciones_proba = lr.predict_proba(X_test)
predicciones_proba

array([[0.8847206 , 0.1152794 ],
       [0.86083744, 0.13916256],
       [0.04160195, 0.95839805],
       [0.77552895, 0.22447105],
       [0.48918718, 0.51081282],
       [0.71250245, 0.28749755],
       [0.22337115, 0.77662885],
       [0.07314229, 0.92685771],
       [0.48850435, 0.51149565],
       [0.22039401, 0.77960599],
       [0.64360498, 0.35639502],
       [0.82327686, 0.17672314],
       [0.67646666, 0.32353334]]
```

Dudas y consultas



Fin Presentación



KIBERNUM