



Universidad Nacional Autónoma de México
Facultad de Ingeniería

Clasificación

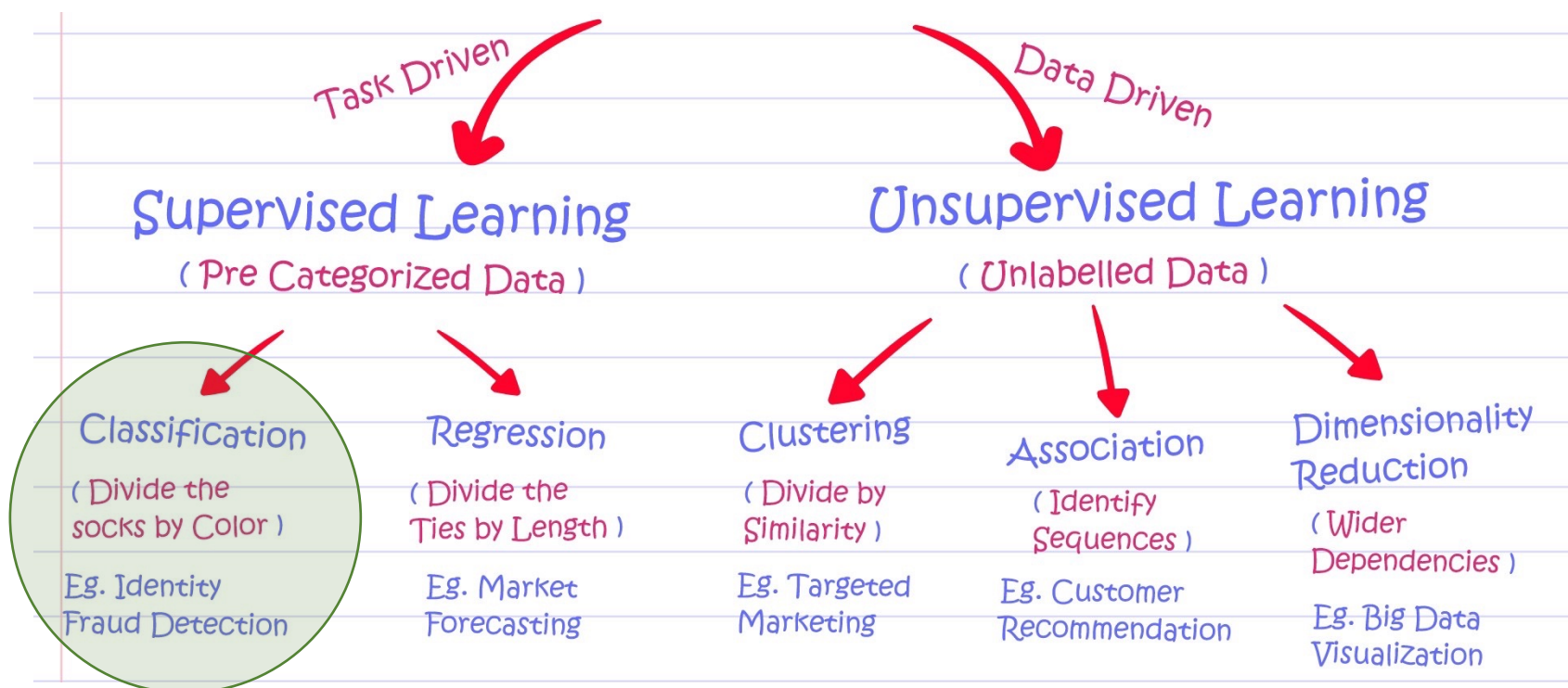
Regresión Logística

Práctica 9

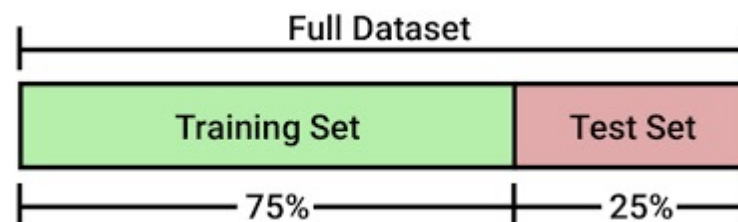
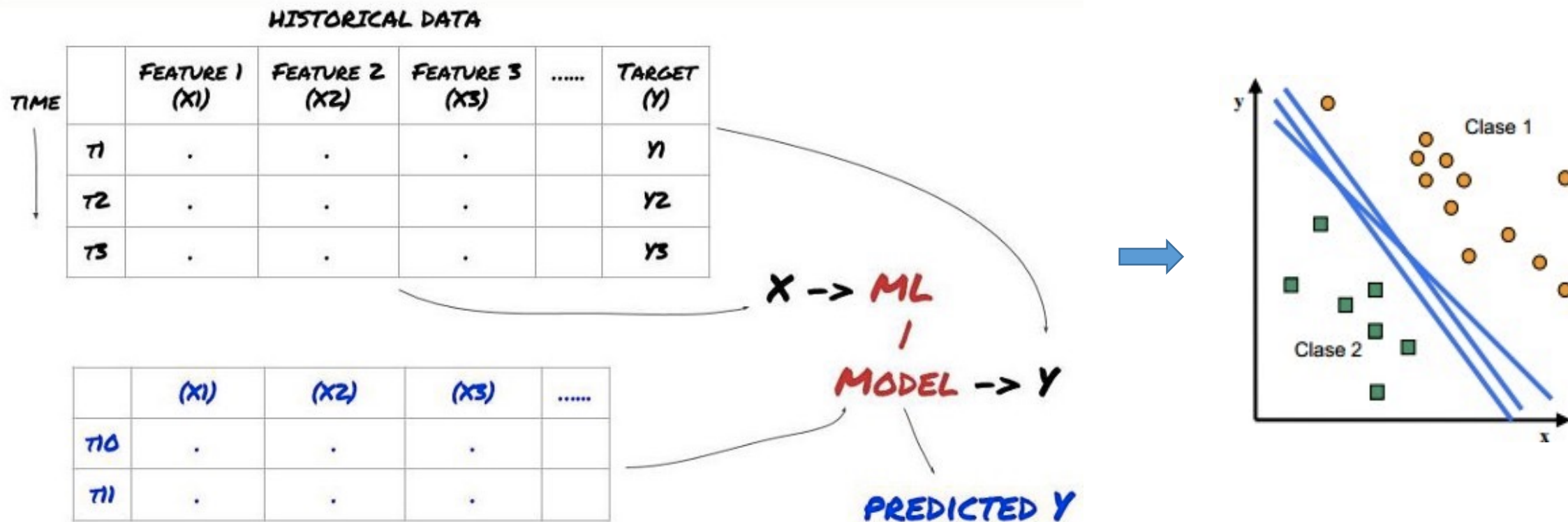
Guillermo Molero-Castillo

guillermo.molero@ingenieria.unam.edu

Noviembre, 2021



Práctica



Fuente de datos

Estudios clínicos a partir de imágenes digitalizadas de pacientes con cáncer de mama de Wisconsin (WDBC, Wisconsin Diagnostic Breast Cancer).

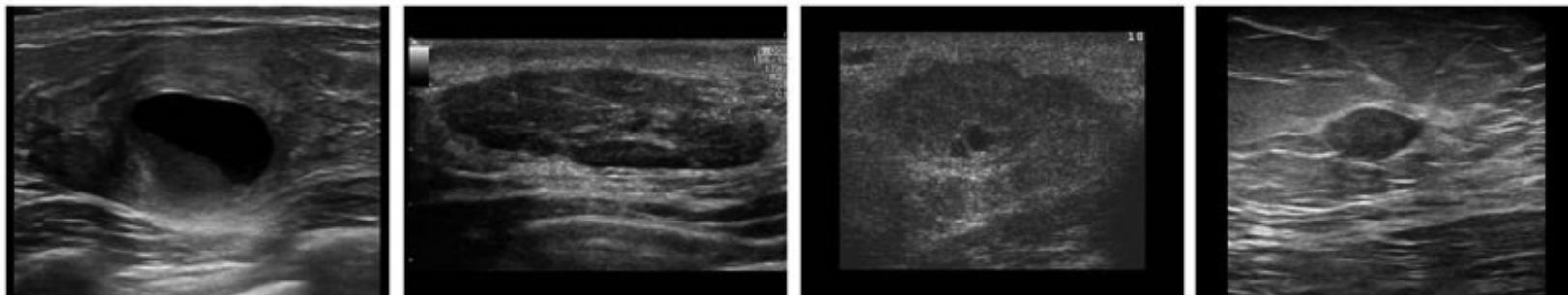
Variable	Descripción	Tipo
ID number	Identifica al paciente	Discreto
Diagnosis	Diagnostico (M=maligno, B=benigno)	Booleano
Radius	Media de las distancias del centro y puntos del perímetro	Continuo
Texture	Desviación estándar de la escala de grises	Continuo
Perimeter	Valor del perímetro del cáncer de mama	Continuo
Area	Valor del área del cáncer de mama	Continuo
Smoothness	Variación de la longitud del radio	Continuo
Compactness	$\text{Perímetro}^2 / \text{Área} - 1$	Continuo
Concavity	Caída o gravedad de las curvas de nivel	Continuo
Concave points	Número de sectores de contorno cóncavo	Continuo
Symmetry	Simetría de la imagen	Continuo
Fractal dimension	"Aproximación de frontera" - 1	Continuo

Fuente: [https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+\(Diagnostic\)](https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+(Diagnostic))

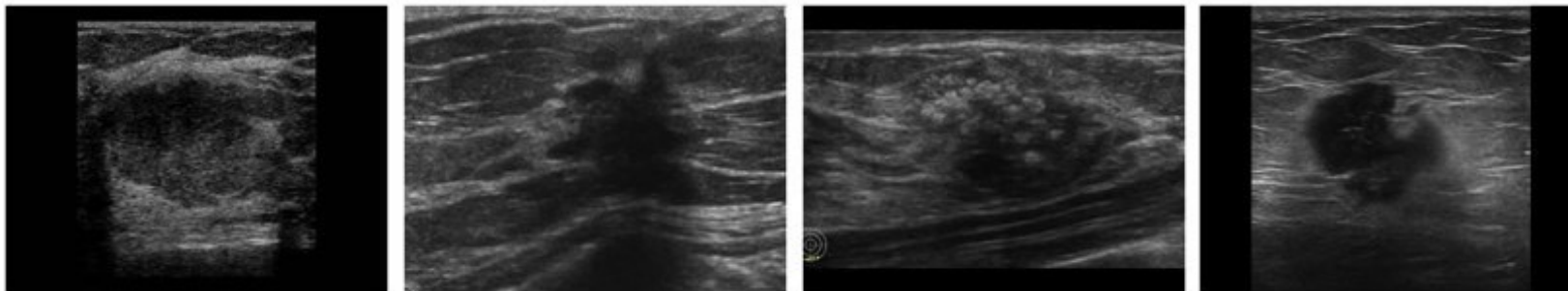
Fuente de datos

Registros clínicos de cáncer de mama a partir de imágenes digitalizadas.

Benigno



Maligno



1. Importar las bibliotecas y los datos

```
▶ import pandas as pd          # Para la manipulación y análisis de datos
import numpy as np            # Para crear vectores y matrices n dimensionales
import matplotlib.pyplot as plt # Para la generación de gráficas a partir de los datos
import seaborn as sns         # Para la visualización de datos basado en matplotlib
%matplotlib inline
```

1. Importar las bibliotecas y los datos

```
BCancer = pd.read_csv('WDBCOriginal.csv')
BCancer
```

Variable dependiente

	IDNumber	Diagnosis	Radius	Texture	Perimeter	Area	Smoothness	Compactness	Concavity	ConcavePoints	Symmetry	FractalDimension
0	P-842302	M	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.30010	0.14710	0.2419	0.07871
1	P-842517	M	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.08690	0.07017	0.1812	0.05667
2	P-84300903	M	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.19740	0.12790	0.2069	0.05999
3	P-84348301	M	11.42	20.38	77.58	386.1	0.14250	0.28390	0.24140	0.10520	0.2597	0.09744
4	P-84358402	M	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.19800	0.10430	0.1809	0.05883
...
564	P-926424	M	21.56	22.39	142.00	1479.0	0.11100	0.11590	0.24390	0.13890	0.1726	0.05623
565	P-926682	M	20.13	28.25	131.20	1261.0	0.09780	0.10340	0.14400	0.09791	0.1752	0.05533
566	P-926954	M	16.60	28.08	108.30	858.1	0.08455	0.10230	0.09251	0.05302	0.1590	0.05648
567	P-927241	M	20.60	29.33	140.10	1265.0	0.11780	0.27700	0.35140	0.15200	0.2397	0.07016
568	P-92751	B	7.76	24.54	47.92	181.0	0.05263	0.04362	0.00000	0.00000	0.1587	0.05884

569 rows x 12 columns

M = Malignant

B = Benign

1. Importar las bibliotecas y los datos

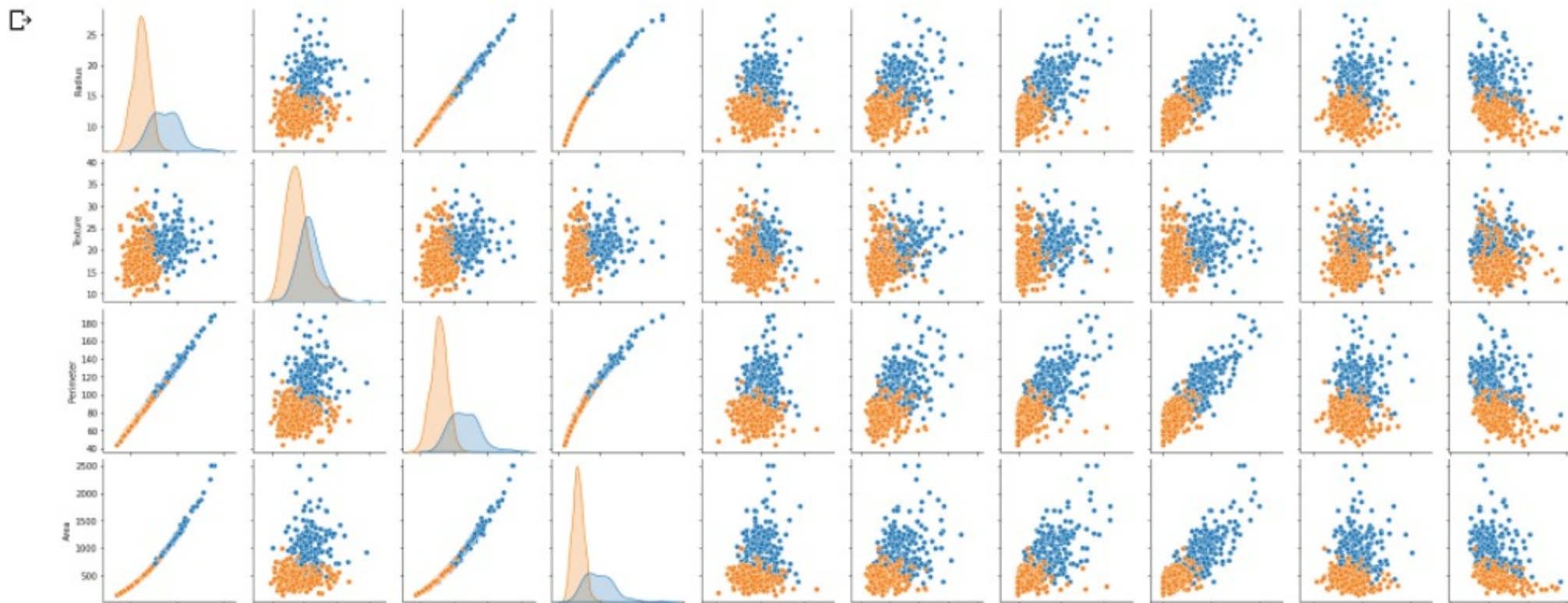
```
▶ print(BCancer.groupby('Diagnosis').size())
```

```
Diagnosis  
B      357  
M      212  
dtype: int64
```


2. Selección de características

Evaluación visual

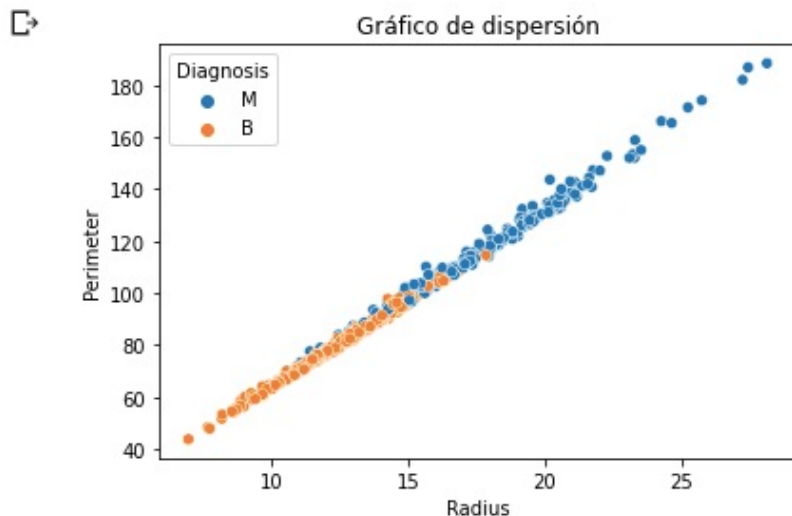
```
sns.pairplot(BCancer, hue='Diagnosis')  
plt.show()
```



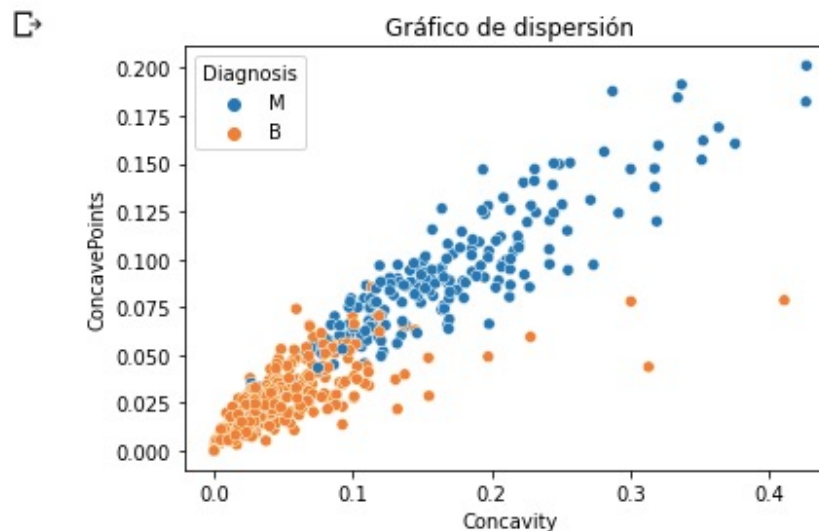
2. Selección de características

Evaluación visual

```
#plt.plot(BCancer['Radius'], BCancer['Perimeter'], 'b+')
sns.scatterplot(x='Radius', y='Perimeter', data=BCancer, hue='Diagnosis')
plt.title('Gráfico de dispersión')
plt.xlabel('Radius')
plt.ylabel('Perimeter')
plt.show()
```



```
#plt.plot(BCancer['Concavity'], BCancer['ConcavePoints'], 'b+')
sns.scatterplot(x='Concavity', y='ConcavePoints', data=BCancer, hue='Diagnosis')
plt.title('Gráfico de dispersión')
plt.xlabel('Concavity')
plt.ylabel('ConcavePoints')
plt.show()
```



2. Selección de características

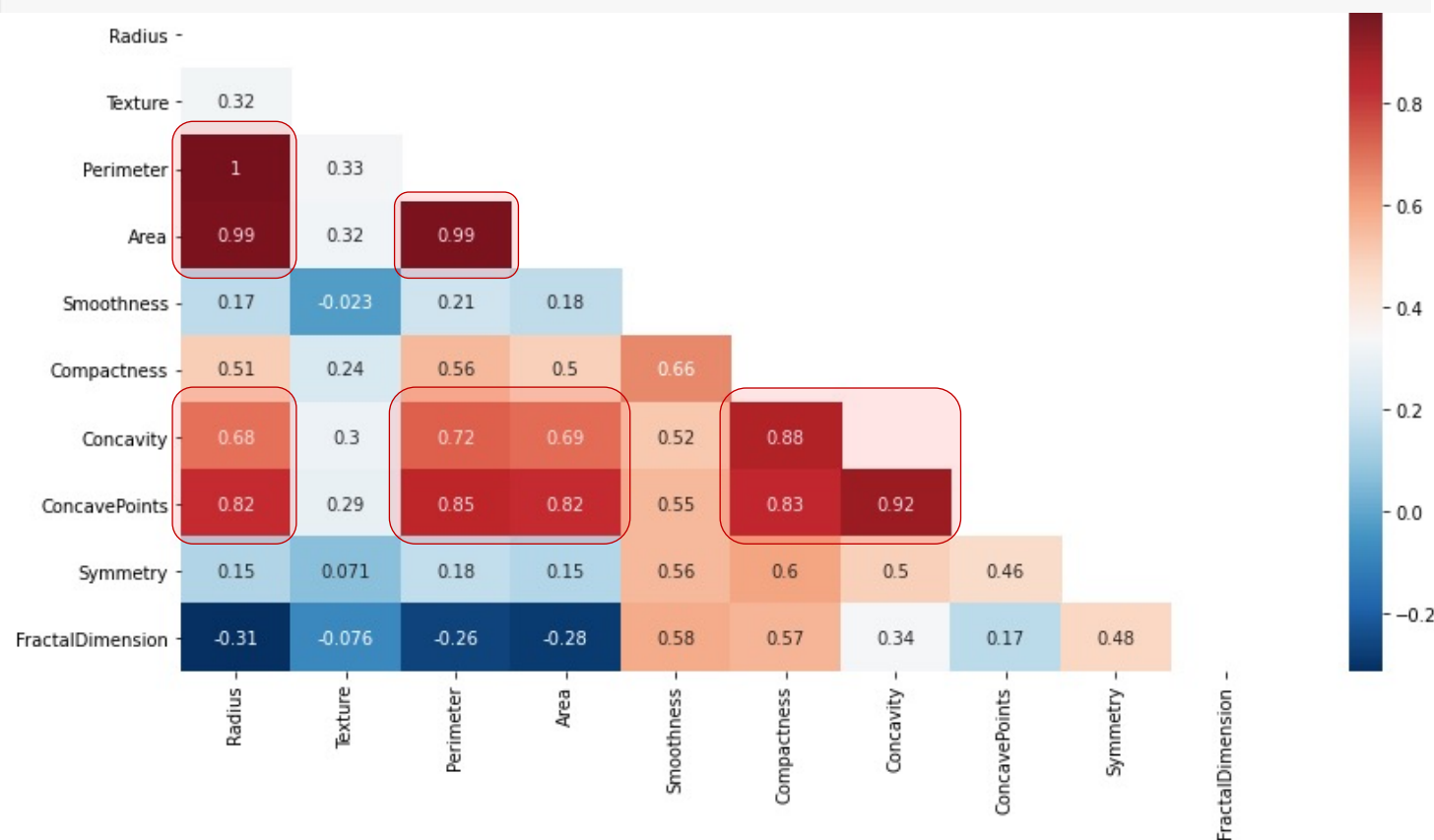
Matriz de correlaciones

```
CorrBCancer = BCancer.corr(method='pearson')
CorrBCancer
```

	Radius	Texture	Perimeter	Area	Smoothness	Compactness	Concavity	ConcavePoints	Symmetry	FractalDimension
Radius	1.000000	0.323782	0.997855	0.987357	0.170581	0.506124	0.676764	0.822529	0.147741	-0.311631
Texture	0.323782	1.000000	0.329533	0.321086	-0.023389	0.236702	0.302418	0.293464	0.071401	-0.076437
Perimeter	0.997855	0.329533	1.000000	0.986507	0.207278	0.556936	0.716136	0.850977	0.183027	-0.261477
Area	0.987357	0.321086	0.986507	1.000000	0.177028	0.498502	0.685983	0.823269	0.151293	-0.283110
Smoothness	0.170581	-0.023389	0.207278	0.177028	1.000000	0.659123	0.521984	0.553695	0.557775	0.584792
Compactness	0.506124	0.236702	0.556936	0.498502	0.659123	1.000000	0.883121	0.831135	0.602641	0.565369
Concavity	0.676764	0.302418	0.716136	0.685983	0.521984	0.883121	1.000000	0.921391	0.500667	0.336783
ConcavePoints	0.822529	0.293464	0.850977	0.823269	0.553695	0.831135	0.921391	1.000000	0.462497	0.166917
Symmetry	0.147741	0.071401	0.183027	0.151293	0.557775	0.602641	0.500667	0.462497	1.000000	0.479921
FractalDimension	-0.311631	-0.076437	-0.261477	-0.283110	0.584792	0.565369	0.336783	0.166917	0.479921	1.000000

2. Selección de características

```
plt.figure(figsize=(14,7))
MatrizInf = np.triu(BCancer.corr())
sns.heatmap(BCancer.corr(), cmap='RdBu_r', annot=True, mask=MatrizInf)
plt.show()
```



Variables seleccionadas:

- 1) Textura [Posición 3]
- 2) Area [Posición 5]
- 3) Smoothness [Pos. 6]
- 4) Compactness [Pos. 7]
- 5) Symmetry [Posición 10]
- 6) FractalDimension [Pos. 11]

3. Definición de variables predictoras (X) y variable clase (Y)

```
BCancer = BCancer.replace({'M': 0, 'B': 1})
BCancer
```

	IDNumber	Diagnosis	Radius	Texture	Perimeter	Area	Smoothness	Compactness	Concavity	ConcavePoints	Symmetry	FractalDimension
0	P-842302	0	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.30010	0.14710	0.2419	0.07871
1	P-842517	0	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.08690	0.07017	0.1812	0.05667
2	P-84300903	0	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.19740	0.12790	0.2069	0.05999
3	P-84348301	0	11.42	20.38	77.58	386.1	0.14250	0.28390	0.24140	0.10520	0.2597	0.09744
4	P-84358402	0	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.19800	0.10430	0.1809	0.05883
...
564	P-926424	0	21.56	22.39	142.00	1479.0	0.11100	0.11590	0.24390	0.13890	0.1726	0.05623
565	P-926682	0	20.13	28.25	131.20	1261.0	0.09780	0.10340	0.14400	0.09791	0.1752	0.05533
566	P-926954	0	16.60	28.08						0.05302	0.1590	0.05648
567	P-927241	0	20.60	29.33						0.15200	0.2397	0.07016
568	P-92751	1	7.76	24.54						0.00000	0.1587	0.05884

569 rows x 12 columns

```
print(BCancer.groupby('Diagnosis').size())

Diagnosis
0      212
1      357
dtype: int64
```

3. Definición de variables predictoras (X) y variable clase (Y)

```
#Variables predictoras
X = np.array(BCancer[['Texture', 'Area', 'Smoothness', 'Compactness', 'Symmetry', 'FractalDimension']])
#X = BCancer.iloc[:, [3, 5, 6, 7, 10, 11]].values #iloc para seleccionar filas y columnas según su posición
pd.DataFrame(X)
```

```
0      10.38    1001.0    0.11840    0.27760    0.2419    0.07871
1      17.77    1326.0    0.08474    0.07864    0.1812    0.05667
2      21.25    1203.0    0.10960    0.15990    0.2069    0.05999
3      20.38     386.1    0.14250    0.28390    0.2597    0.09744
4      14.34    1297.0    0.10030    0.13280    0.1809    0.05883
...
564    22.39    1479.0    0.11100    0.11590    0.1726    0.05623
565    28.25    1261.0    0.09780    0.10340    0.1752    0.05533
566    28.08     858.1    0.08455    0.10230    0.1590    0.05648
567    29.33    1265.0    0.11780    0.27700    0.2397    0.07016
568    24.54     181.0    0.05263    0.04362    0.1587    0.05884
```

569 rows x 6 columns

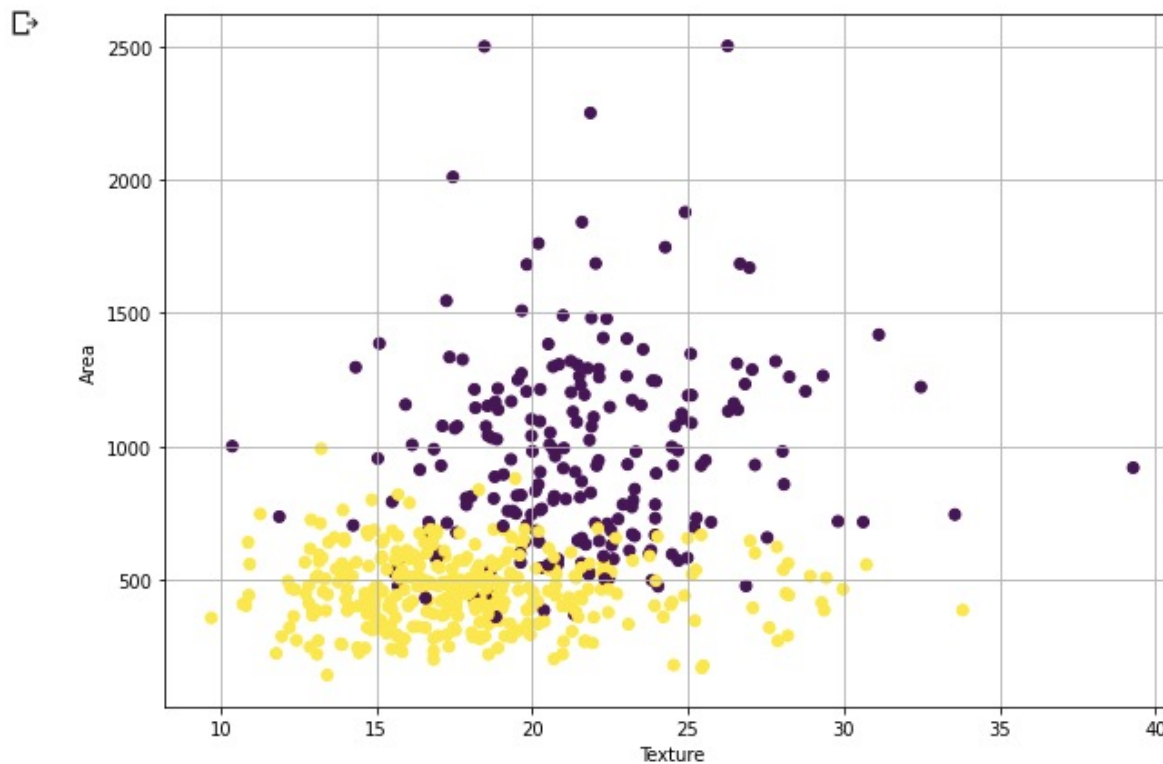
3. Definición de variables predictoras (X) y variable clase (Y)

```
#Variable clase
Y = np.array(BCancer[['Diagnosis']])
pd.DataFrame(Y)
```

```
0
0 0
1 0
2 0
3 0
4 0
...
564 0
565 0
566 0
567 0
568 1
```

569 rows x 1 columns

```
plt.figure(figsize=(10, 7))
plt.scatter(X[:,0], X[:,1], c = BCancer.Diagnosis)
plt.grid()
plt.xlabel('Texture')
plt.ylabel('Area')
plt.show()
```

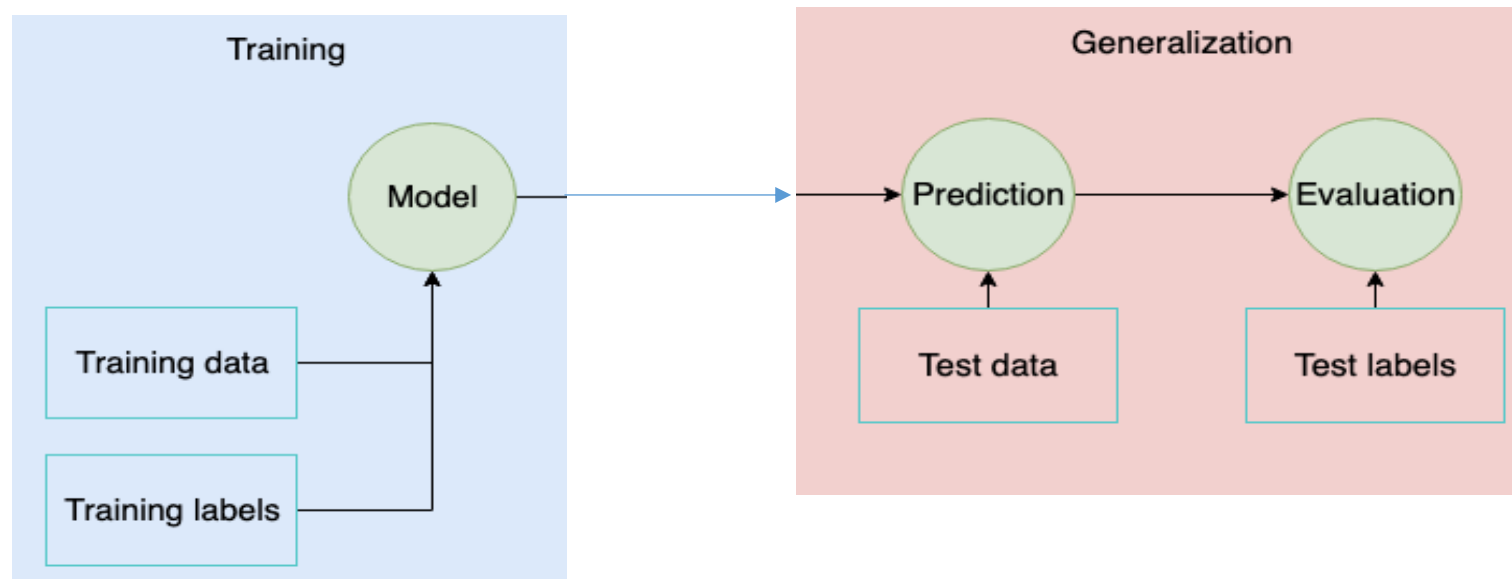


4. Aplicación del algoritmo

```
▶ #Se importan las bibliotecas necesarias para generar el modelo de regresión logística
from sklearn import linear_model
from sklearn import model_selection
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
```


4. Aplicación del algoritmo


Se hace la división de los datos



```
▶ X_train, X_validation, Y_train, Y_validation = model_selection.train_test_split(X, Y,  
                                     test_size = 0.2,  
                                     random_state = 1234,  
                                     shuffle = True)
```


4. Aplicación del algoritmo

Se hace la división de los datos

 `pd.DataFrame(X_train)`

	0	1	2	3	4	5
0	18.22	493.1	0.12180	0.16610	0.1709	0.07253
1	22.44	378.4	0.09566	0.08194	0.2030	0.06552
2	20.76	480.4	0.09933	0.12090	0.1735	0.07070
3	23.84	499.0	0.11220	0.12620	0.1905	0.06590
4	18.32	340.9	0.08142	0.04462	0.2372	0.05768
...
450	15.18	587.4	0.09516	0.07688	0.2110	0.05853
451	15.10	1386.0	0.10010	0.15150	0.1973	0.06183
452	18.60	481.9	0.09965	0.10580	0.1925	0.06373
453	18.70	1033.0	0.11480	0.14850	0.2092	0.06310
454	13.78	492.1	0.09667	0.08393	0.1638	0.06100

455 rows x 6 columns

 `pd.DataFrame(Y_train)`

	0
0	1
1	1
2	1
3	0
4	1
...	...
450	1
451	0
452	1
453	0
454	1

455 rows x 1 columns

4. Aplicación del algoritmo

Se entrena el modelo



#Se entrena el modelo a partir de los datos de entrada

```
Clasificacion = linear_model.LogisticRegression()
```

```
Clasificacion.fit(X_train, Y_train)
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/utils/validation.py:760: Data:
y = column_or_1d(y, warn=True)
```

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                    intercept_scaling=1, l1_ratio=None, max_iter=100,
                    multi_class='auto', n_jobs=None, penalty='l2',
                    random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                    warm_start=False)
```

4. Aplicación del algoritmo

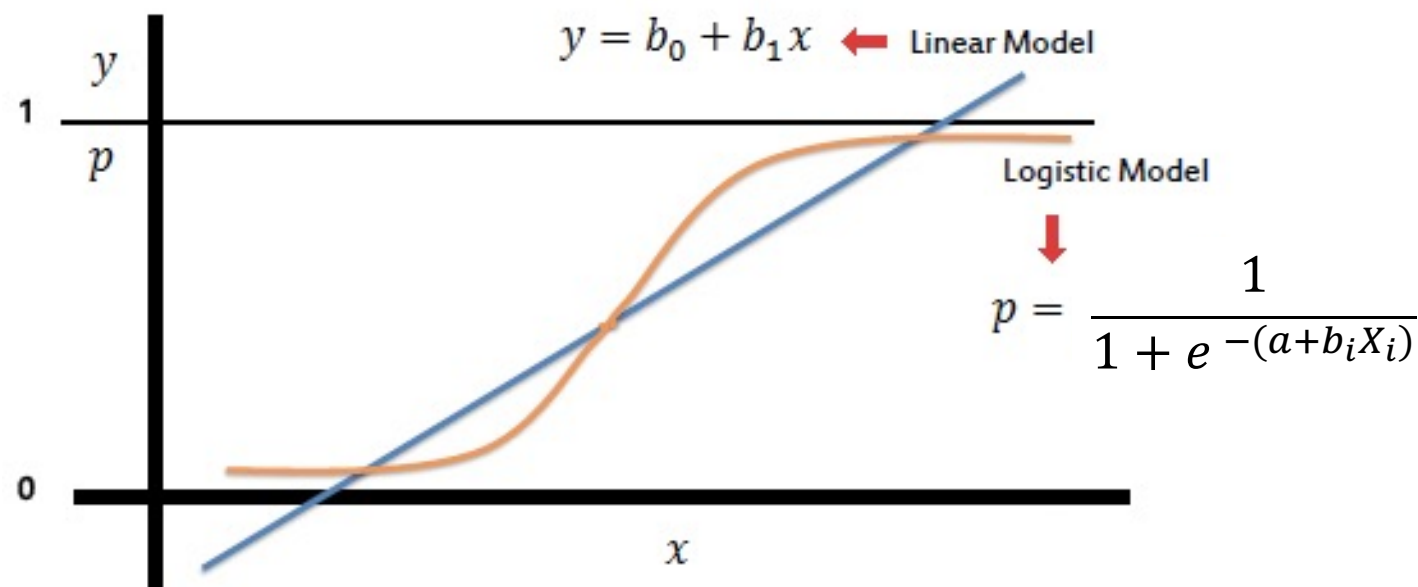
Se generan las probabilidades

```
#Predicciones probabilísticas
Probabilidad = Clasificacion.predict_proba(X_train)
pd.DataFrame(Probabilidad)
```

	0	1
0	0.095368	0.904632
1	0.051662	0.948338
2	0.116534	0.883466
3	0.235630	0.764370
4	0.014660	0.985340
...
450	0.118724	0.881276
451	0.999155	0.000845
452	0.079253	0.920747
453	0.979264	0.020736
454	0.033240	0.966760

455 rows x 2 columns

Diagnóstico



4. Aplicación del algoritmo

Se generan las predicciones de clasificación

```
#Predicciones probabilísticas de los datos de prueba
Probabilidad = Clasificacion.predict_proba(X_validation)
pd.DataFrame(Probabilidad)
```

	0	1
0	0.050099	9.499011e-01
1	0.003135	9.968647e-01
2	0.057000	9.430004e-01
3	0.011637	9.883630e-01
4	0.065728	9.342722e-01
...
109	0.057255	9.427452e-01
110	0.990748	9.252494e-03
111	0.066344	9.336558e-01
112	0.186568	8.134320e-01
113	1.000000	3.283193e-10

114 rows x 2 columns

```
#Predicciones con clasificación final
Predicciones = Clasificacion.predict(X_validation)
pd.DataFrame(Predicciones)
```

	0
0	1
1	1
2	1
3	1
4	1

```
#Se calcula el exactitud promedio de la validación
Clasificacion.score(X_validation, Y_validation)
```

0.9385964912280702

112	1
113	0

114 rows x 1 columns

5. Validación del modelo

```
#Matriz de clasificación
Y_Clasificacion = Clasificacion.predict(X_validation)
Matriz_Clasificacion = pd.crosstab(Y_validation.ravel(),
                                   Y_Clasificacion,
                                   rownames=[ 'Real' ],
                                   colnames=[ 'Clasificación' ])

Matriz_Clasificacion
```

↗

Clasificación	0	1
Real		
0	39	6
1	1	68

5. Validación del modelo



#Reporte de la clasificación

```
print("Exactitud", Clasificacion.score(X_validation, Y_validation))
```

```
print(classification_report(Y_validation, Y_Clasificacion))
```

Exactitud 0.9385964912280702

	precision	recall	f1-score	support
0	0.97	0.87	0.92	45
1	0.92	0.99	0.95	69
accuracy			0.94	114
macro avg	0.95	0.93	0.93	114
weighted avg	0.94	0.94	0.94	114

5. Validación del modelo

Clasificación	0	1
Real		
0	39	6
1	1	68

Exactitud 0.9385964912280702				
	precision	recall	f1-score	support
0	0.97	0.87	0.92	45
1	0.92	0.99	0.95	69
accuracy			0.94	114
macro avg	0.95	0.93	0.93	114
weighted avg	0.94	0.94	0.94	114

$$\text{Exactitud} = \frac{VP+VN}{VP+VN+FP+FN} = \frac{68+39}{68+39+1+6} = \frac{107}{114} = \mathbf{0.938}$$

$$\text{Precisión} = \frac{VP}{VP+FP} = \frac{68}{68+6} = \frac{68}{74} = \mathbf{0.918}$$

$$\text{Tasa de error} = \frac{FP+FN}{VP+VN+FP+FN} = \frac{1+6}{68+39+1+6} = \frac{7}{114} = \mathbf{0.06}$$

$$\text{Sensibilidad} = \frac{VP}{VP+FN} = \frac{68}{68+1} = \frac{68}{69} = \mathbf{0.985}$$

$$\text{Especificidad} = \frac{VN}{VN+FP} = \frac{39}{39+6} = \frac{39}{45} = \mathbf{0.866}$$

6. Modelo de clasificación

```
#Ecuación del modelo
print("Intercept:", Clasificacion.intercept_)
print('Coeficientes: \n', Clasificacion.coef_)
```

Intercept: [12.0257237]

Coeficientes:

[[-0.19554751 -0.01115866 -0.70751733 -2.59203115 -1.02579301 -0.25791963]]

$$Prob = \frac{1}{1 + e^{-(a+bX)}}$$

$$a + bX = 12.025 - 0.195Texture - 0.011Area - 0.707Smoothness - 2.592Compactness - 1.025Symmetry - 0.257FractalDimension$$

6. Modelo de clasificación

```
#Paciente P-842302 (1) -Tumor Maligno-
PacienteID1 = pd.DataFrame({'Texture': [10.38],
                             'Area': [1001.0],
                             'Smoothness': [0.11840],
                             'Compactness': [0.27760],
                             'Symmetry': [0.2419],
                             'FractalDimension': [0.07871]})

Clasificacion.predict(PacienteID1)
```

```
array([0])
```

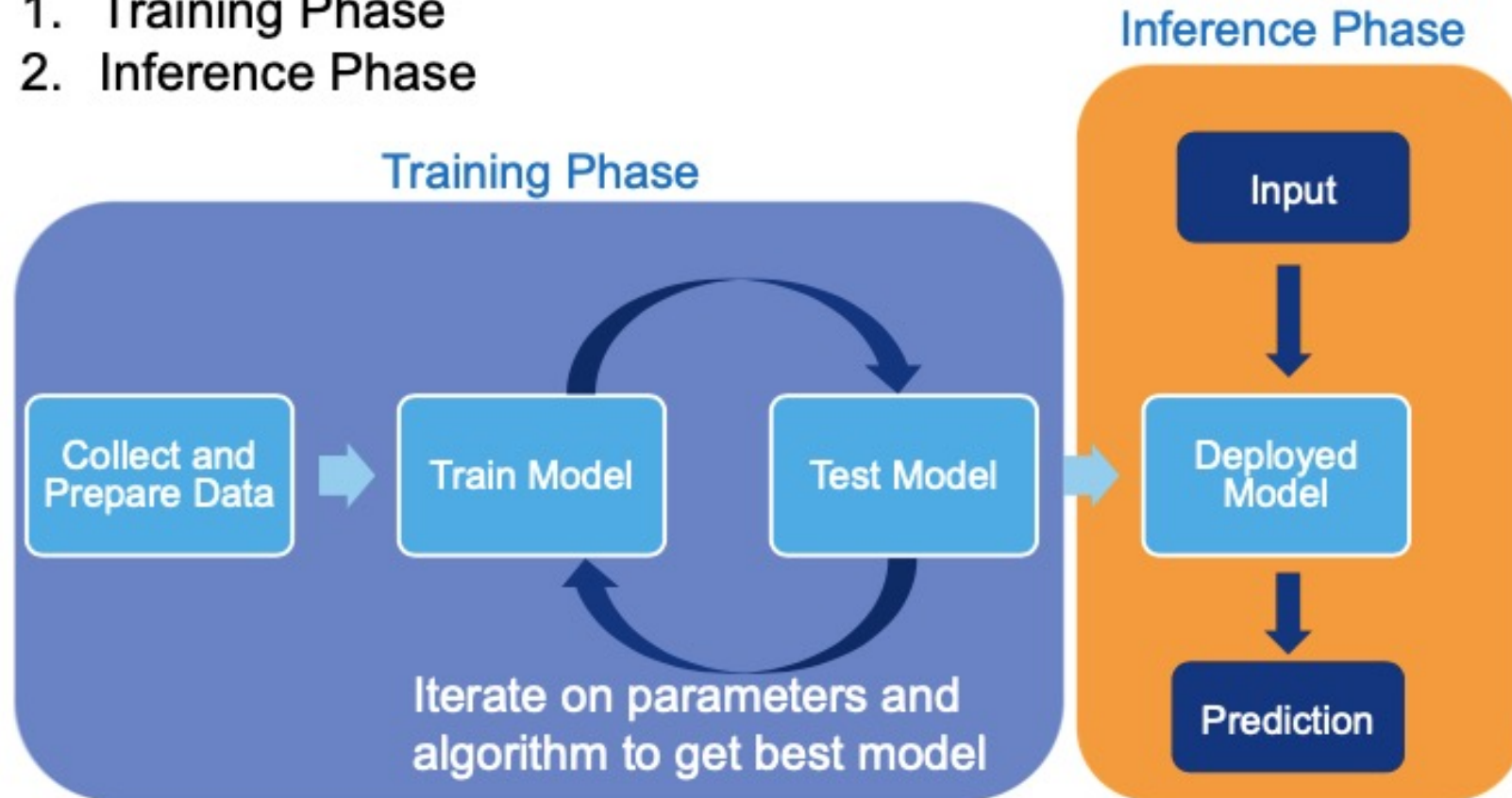
```
#Paciente P-92751 (569) -Tumor Benigno-
PacienteID2 = pd.DataFrame({'Texture': [24.54],
                             'Area': [181.0],
                             'Smoothness': [0.05263],
                             'Compactness': [0.04362],
                             'Symmetry': [0.1587],
                             'FractalDimension': [0.05884]})

Clasificacion.predict(PacienteID2)
```

```
array([1])
```

Sistema de inferencia basado en un modelo de regresión logística

1. Training Phase
2. Inference Phase



Sistemas de inferencia basadas en modelos de predicción

ID Paciente	<input type="text"/>		
Texture	<input type="text"/>	Compactness	<input type="text"/>
Area	<input type="text"/>	Symmetry	<input type="text"/>
Smoothness	<input type="text"/>	Fractal Dimension	<input type="text"/>
Diagnosis	<input type="text"/>		