In [1]:

```
%load_ext watermark
%watermark
```

```
2019-05-30T21:34:12+02:00

CPython 3.6.5
IPython 6.4.0

compiler   : GCC 7.2.0
system     : Linux
release    : 5.1.5-arch1-2-ARCH
machine    : x86_64
processor  :
CPU cores  : 4
interpreter: 64bit
```

Evaluación de Modelos de Clasificación

En este apartado evaluaremos la calidad del modelo, utilizando técnicas como la clasificación binaria. Como en el apartado de regresión logística, emplearemos el dataset breast_cancer

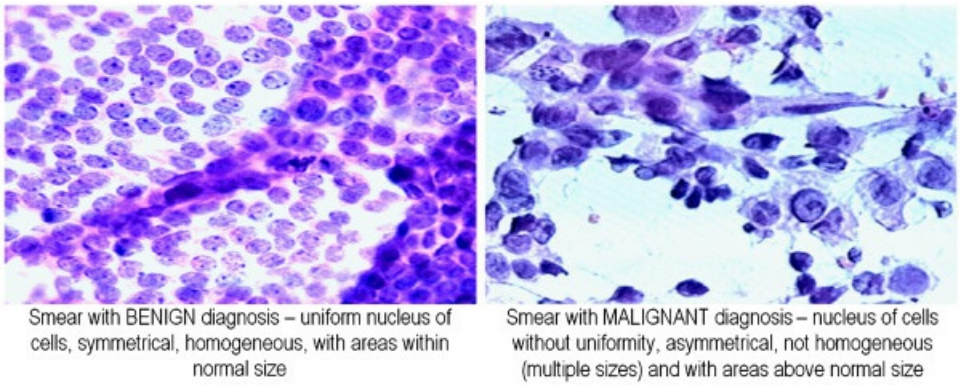
In [2]:

```
%matplotlib inline
import matplotlib.pyplot as plt

import pandas as pd
import numpy as np
from sklearn import datasets
```

In [3]:

```
from IPython.display import Image

Image("../../RESOURCES/breast_cancer.jpeg")
```

Out[3]:



Smear with BENIGN diagnosis – uniform nucleus of cells, symmetrical, homogeneous, with areas within normal size

Smear with MALIGNANT diagnosis – nucleus of cells without uniformity, asymmetrical, not homogeneous (multiple sizes) and with areas above normal size

Ingesta de Datos

In [4]:

```
cancer_datos = datasets.load_breast_cancer()

cancer_df = pd.DataFrame(cancer_datos["data"],
                         columns=cancer_datos["feature_names"]
                        )

cancer_df["objetivo"] = cancer_datos.target
cancer_df["objetivo"] = cancer_df["objetivo"].replace({0:1, 1:0})
```

In [5]:

```
cancer_df["objetivo"].value_counts(True)
```

Out[5]:

```
0    0.627417
1    0.372583
Name: objetivo, dtype: float64
```

In [6]:

```
cancer_df.head()
```

Out[6]:

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mean symmetry	mean fractal dimension	...	worst texture	per
0	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.14710	0.2419	0.07871	...	17.33	184
1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0.07017	0.1812	0.05667	...	23.41	158
2	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.12790	0.2069	0.05999	...	25.53	152
3	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0.10520	0.2597	0.09744	...	26.50	98.
4	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	0.10430	0.1809	0.05883	...	16.67	152

5 rows × 31 columns

Creción de Modelo de Regresión Logística

In [7]:

```
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn import metrics
```

In [8]:

```
X = cancer_df[cancer_datos.feature_names]
y = cancer_df["objetivo"]

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
```

In [9]:

```
modelo = LogisticRegression()

modelo.fit(X_train, y_train)

predicciones = modelo.predict(X_test)
clases_reales = y_test
#predecimos las probabilidades
predicciones_probabilidades = modelo.predict_proba(X_test)
```

```
/opt/anaconda3/lib/python3.6/site-packages/sklearn/linear_model/logistic.py:433: FutureWarning: De
fault solver will be changed to 'lbfgs' in 0.22. Specify a solver to silence this warning.
  FutureWarning)
```

Implementaremos una función que devuelva una lista para cada observación entre la clase real y la clase predicho

In [10]:

```
def tupla_clase_prediccion(y_real, y_pred):
    return list(zip(y_real, y_pred))

tupla_clase_prediccion(clases_reales, predicciones)[:10]
```

Out[10]:

```
[(0, 0),
 (1, 1),
 (1, 1),
 (0, 0),
 (0, 0),
 (1, 1),
```

```
 (1, 1),
 (1, 1),
 (0, 0),
 (0, 0)]
```

Conceptos de Clasificación binaria

En clasificación binaria, tenemos el concepto de casos negativos (clase 0, en el caso del dataset de cancer de mama serian los casos donde el cancer es benigno) y casos positivos (clase 1, en el caso del dataset de cancer de mama serían los casos donde el cancer es maligno). Esto nos lleva a cuatro tipos de posibilidades u observaciones posibles:

- Verdaderos Positivos(True positives), serían las imágenes con un cancer maligno que se detectan como cancer maligno.
- Falsos Positivos (False positives), serían los cánceres benignos que se detectan como un cancer maligno.
- Verdaderos Negativos(True Negatives), serían los canceres benignos que se clasifican como cánceres benignos.
- Falsos Negativos(False Negatives), serían los canceres malignos que se clasifican como cánceres benignos.

Implementaremos una serie de funciones encargadas de calcular los valores correspondientes para los cuatro tipo de observaciones realizadas.

In [11]:

```
def VP(clases_reales, predicciones):
    par_clase_prediccion = tupla_clase_prediccion(clases_reales, predicciones)
    return len([obs for obs in par_clase_prediccion if obs[0]==1 and obs[1]==1])

def VN(clases_reales, predicciones):
    par_clase_prediccion = tupla_clase_prediccion(clases_reales, predicciones)
    return len([obs for obs in par_clase_prediccion if obs[0]==0 and obs[1]==0])

def FP(clases_reales, predicciones):
    par_clase_prediccion = tupla_clase_prediccion(clases_reales, predicciones)
    return len([obs for obs in par_clase_prediccion if obs[0]==0 and obs[1]==1])

def FN(clases_reales, predicciones):
    par_clase_prediccion = tupla_clase_prediccion(clases_reales, predicciones)
    return len([obs for obs in par_clase_prediccion if obs[0]==1 and obs[1]==0])


print("""
Verdaderos Positivos: {}
Verdaderos Negativos: {}
Falsos Positivos: {}
Falsos Negativos: {}
""".format(
    VP(clases_reales, predicciones),
    VN(clases_reales, predicciones),
    FP(clases_reales, predicciones),
    FN(clases_reales, predicciones)
))
```

```
Verdaderos Positivos: 59
Verdaderos Negativos: 106
Falsos Positivos: 2
Falsos Negativos: 4
```

Ratios de clasificación

Exactitud (Accuracy)

La exactitud es una medida general de como se comporta el modelo, mide simplemente el porcentaje de casos que se han clasificado correctamente.

$$Exactitud=\frac{Número~de~observaciones~correctamente~clasificadas}{Número~de~observaciones~totales}= \frac{VP+VN}{VP+VN+FP+FN}$$

In [12]:

```
def exactitud(clases_reales, predicciones):
    vp = VP(clases_reales, predicciones)
    vn = VN(clases_reales, predicciones)
    return (vp+vn) / len(clases_reales)
```

```
exactitud(clases_reales, predicciones)
```

Out[12]:

```
0.9649122807017544
```

In [13]:

```
from sklearn import metrics
metrics.accuracy_score(clases_reales, predicciones)
```

Out[13]:

```
0.9649122807017544
```

Precisión (Precission)

La precisión indica la habilidad del modelo para clasificar como positivos los casos que son positivos.

$$Precisión=\frac{Número~de~observaciones~positivas~correctamente~clasificadas}{Número~de~observaciones~clasificadas~como~positivas}= \frac{VP}{VP+FP}$$

In [14]:

```
def precision(clases_reales, predicciones):
    vp = VP(clases_reales, predicciones)
    fp = FP(clases_reales, predicciones)
    return vp / (vp+fp)

precision(clases_reales, predicciones)
```

Out[14]:

```
0.9672131147540983
```

In [15]:

```
metrics.average_precision_score(clases_reales, predicciones)
```

Out[15]:

```
0.9291945711272717
```

Exhaustividad o sensibilidad(Recall o True Positive Rate)

La sensibilidad nos da una medida de la habilidad del modelo para encontrar todos los casos positivos. La sensibilidad se mide en función de una clase.

$$Sensibilidad=\frac{Número~de~observaciones~positivas~clasificadas~como~positivas}{Número~de~observaciones~positivas~totales}= \frac{VP}{VP+FN}$$

In [16]:

```
def sensibilidad(clases_reales, predicciones):
    vp = VP(clases_reales, predicciones)
    fn = FN(clases_reales, predicciones)
    return vp / (vp+fn)

sensibilidad(clases_reales, predicciones)
```

Out[16]:

```
0.9365079365079365
```

In [17]:

```
metrics.recall_score(clases_reales, predicciones)
```

Out[17]:

```
0.9365079365079365
```

Matriz de confusion

La matriz de confusión es una forma muy sencilla de comparar como ha clasificado cada observación el modelo.

In [18]:

```
from sklearn.metrics import confusion_matrix
```

```
from sklearn.metrics import confusion_matrix
confusion_matrix(clases_reales, predicciones)
```

Out[18]:

```
array([[106,   2],
       [  4,  59]])
```

Puntuación F1 (F1 score)

La puntuación F1 es una media ponderada entre la sensibilidad (que intenta obtener cuantos mas verdaderos positivos independientemente de los falsos positivos) y la precisión (que intenta obtener solo verdaderos positivos que sean casos claros para limitar los falsos positivos).

La puntuación F1 se define como la media harmónica de la precisión y la sensibilidad:

$$F1=2*\frac{1}{\frac{1}{precisión}+\frac{1}{sensibilidad}}=2*\frac{precisión*sensibilidad}{precisión+sensibilidad}$$

In [19]:

```
def puntuacion_f1(clases_reales, predicciones):
    precision_preds = precision(clases_reales, predicciones)
    sensibilidad_preds = sensibilidad(clases_reales, predicciones)
    return 2*(precision_preds*sensibilidad_preds)/(precision_preds+sensibilidad_preds)

puntuacion_f1(clases_reales, predicciones)
```

Out[19]:

```
0.9516129032258064
```

In [20]:

```
metrics.f1_score(clases_reales, predicciones)
```

Out[20]:

```
0.9516129032258064
```

Ratio de Falsos Positivos (Ratio de Falsa Alarma o FPR)

El ratio de falsos positivos nos da una medida de las probabilidades de nuestro modelo de asignar una clase positiva a un caso negativo.

Se define como:

$$FPR=\frac{Número~de~observaciones~negativas~clasificadas~como~positivas}{Número~de~observaciones~negativas}=\frac{FP}{FP+TN}$$

In [21]:

```
def fpr(clases_reales, predicciones):
    return (FP(clases_reales, predicciones) / (
              FP(clases_reales, predicciones) + VN(clases_reales, predicciones)
              )
            )
fpr(clases_reales, predicciones)
```

Out[21]:

```
0.018518518518518517
```