In [1]:

```
%load_ext watermark
%watermark
%matplotlib inline
```

```
2019-05-30T21:33:44+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
```

Regresion logistica

Teoría

Variables Categóricas.

Como hemos estudidado anteriormente, el aprendizaje en machine Learning se divide en aprendizaje supervisado y aprendizaje no supervisado. A su vez, el aprendizaje supervisado contempla dos tipos de variables: Las variables continuas y las variables categóricas.

Los problemas de clasificación son muy extensos, desde clasificación de imagenes a la clasificación de carácteres escritos.

¿Que es la Regresión Logística?

En estadística, la regresión logística es un tipo de análisis de regresión utilizado para predecir el resultado de una variable categórica (una variable que puede adoptar un número limitado de categorías) en función de las variables independientes o predictoras. Es útil para modelar la probabilidad de un evento ocurriendo como función de otros factores. El análisis de regresión logística se enmarca en el conjunto de Modelos Lineales Generalizados (GLM por sus siglas en inglés) que usa como función de enlace la función logit. Las probabilidades que describen el posible resultado de un único ensayo se modelan, como una función de variables explicativas, utilizando una función logística.

La regresión logística es usada extensamente en las ciencias médicas y sociales. Otros nombres para regresión logística usados en varias áreas de aplicación incluyen modelo logístico, modelo logit, y clasificador de máxima entropía.

La regresión logística se puede utilizar para resolver problemas donde queremos clasificar,ya que la principal diferencia entre la regresión lineal y la logística es que en estas últimas las variables objetivo que trata son probabilidades de una clase (*Clase A o Clase B* o valores booleanos).

Esto traduce a esu funcion lineal a una funcion donde el resultado simpre estara en cero.

Implementacion

Ingesta de datos

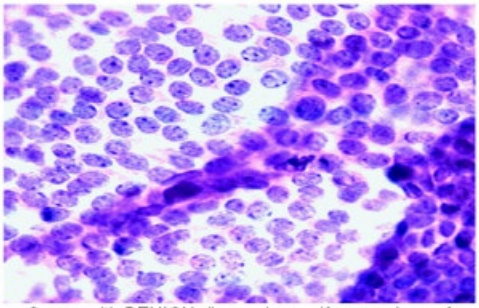
Para este ejemplo vamos a ultilizar el (**Wisconsin Breast Cancer Dataset**). Es un dataset de imagenes de celulas ibtenidas de analisis de personas que sufren un posible cancer de mama.

Las imagenes tiene el siguiente aspecto:

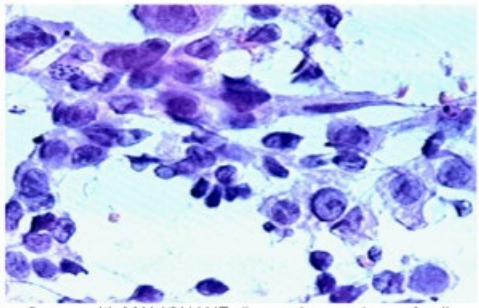
In [2]:

```
from IPython.display import Image

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

Out[2]:



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

In [3]:

```
%matplotlib inline
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
from sklearn import datasets
```

In [4]:

```
cancer_datos = datasets.load_breast_cancer()
cancer_datos.keys()
```

Out[4]:

```
dict_keys(['data', 'target', 'target_names', 'DESCR', 'feature_names', 'filename'])
```

In [5]:

```
print(cancer_datos["DESCR"])
```

```
.. _breast_cancer_dataset:

Breast cancer wisconsin (diagnostic) dataset
--------------------------------------------

**Data Set Characteristics:**

    :Number of Instances: 569

    :Number of Attributes: 30 numeric, predictive attributes and the class

    :Attribute Information:
        - radius (mean of distances from center to points on the perimeter)
        - texture (standard deviation of gray-scale values)
        - perimeter
        - area
        - smoothness (local variation in radius lengths)
        - compactness (perimeter^2 / area - 1.0)
        - concavity (severity of concave portions of the contour)
        - concave points (number of concave portions of the contour)
        - symmetry
        - fractal dimension ("coastline approximation" - 1)

        The mean, standard error, and "worst" or largest (mean of the three
        largest values) of these features were computed for each image,
        resulting in 30 features.  For instance, field 3 is Mean Radius, field
        13 is Radius SE, field 23 is Worst Radius.

        - class:
                - WDBC-Malignant
                - WDBC-Benign

    :Summary Statistics:

    ===================================== ====== ======
                                           Min    Max
    ===================================== ====== ======
    radius (mean):                        6.981  28.11
    texture (mean):                       9.71   39.28
    perimeter (mean):                     43.79  188.5
    area (mean):                          143.5  2501.0
```

```
    area (mean):                            143.5  2501.0
    smoothness (mean):                      0.053  0.163
    compactness (mean):                     0.019  0.345
    concavity (mean):                       0.0    0.427
    concave points (mean):                  0.0    0.201
    symmetry (mean):                        0.106  0.304
    fractal dimension (mean):               0.05   0.097
    radius (standard error):                0.112  2.873
    texture (standard error):               0.36   4.885
    perimeter (standard error):             0.757  21.98
    area (standard error):                  6.802  542.2
    smoothness (standard error):            0.002  0.031
    compactness (standard error):           0.002  0.135
    concavity (standard error):             0.0    0.396
    concave points (standard error):        0.0    0.053
    symmetry (standard error):              0.008  0.079
    fractal dimension (standard error):     0.001  0.03
    radius (worst):                         7.93   36.04
    texture (worst):                        12.02  49.54
    perimeter (worst):                      50.41  251.2
    area (worst):                           185.2  4254.0
    smoothness (worst):                     0.071  0.223
    compactness (worst):                    0.027  1.058
    concavity (worst):                      0.0    1.252
    concave points (worst):                 0.0    0.291
    symmetry (worst):                       0.156  0.664
    fractal dimension (worst):              0.055  0.208
    ========================================== ====== ======

    :Missing Attribute Values: None

    :Class Distribution: 212 - Malignant, 357 - Benign

    :Creator:  Dr. William H. Wolberg, W. Nick Street, Olvi L. Mangasarian

    :Donor: Nick Street

    :Date: November, 1995

This is a copy of UCI ML Breast Cancer Wisconsin (Diagnostic) datasets.
https://goo.gl/U2Uwz2

Features are computed from a digitized image of a fine needle
aspirate (FNA) of a breast mass.  They describe
characteristics of the cell nuclei present in the image.

Separating plane described above was obtained using
Multisurface Method-Tree (MSM-T) [K. P. Bennett, "Decision Tree
Construction Via Linear Programming." Proceedings of the 4th
Midwest Artificial Intelligence and Cognitive Science Society,
pp. 97-101, 1992], a classification method which uses linear
programming to construct a decision tree.  Relevant features
were selected using an exhaustive search in the space of 1-4
features and 1-3 separating planes.

The actual linear program used to obtain the separating plane
in the 3-dimensional space is that described in:
[K. P. Bennett and O. L. Mangasarian: "Robust Linear
Programming Discrimination of Two Linearly Inseparable Sets",
Optimization Methods and Software 1, 1992, 23-34].

This database is also available through the UW CS ftp server:

ftp ftp.cs.wisc.edu
cd math-prog/cpo-dataset/machine-learn/WDBC/

.. topic:: References

   - W.N. Street, W.H. Wolberg and O.L. Mangasarian. Nuclear feature extraction
     for breast tumor diagnosis. IS&T/SPIE 1993 International Symposium on
     Electronic Imaging: Science and Technology, volume 1905, pages 861-870,
     San Jose, CA, 1993.
   - O.L. Mangasarian, W.N. Street and W.H. Wolberg. Breast cancer diagnosis and
     prognosis via linear programming. Operations Research, 43(4), pages 570-577,
     July-August 1995.
   - W.H. Wolberg, W.N. Street, and O.L. Mangasarian. Machine learning techniques
     to diagnose breast cancer from fine-needle aspirates. Cancer Letters 77 (1994)
     163-171
```

163-171.

In [6]:

```
cancer_datos["target"][:20]
```

Out[6]:

```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1])
```

In [7]:

```
cancer_datos.target_names
```

Out[7]:

```
array(['malignant', 'benign'], dtype='<U9')
```

In [8]:

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

cancer_df["objetivo"] = cancer_datos.target
```

El dataset contiene los valores medios de ciertis parametros del núcleo de las celulas mostradas en las iágenes, así como dichos valores para la celula con características más preocupantes

In [9]:

```
cancer_df.shape
```

Out[9]:

```
(569, 31)
```

In [10]:

```
cancer_df.head()
```

Out[10]:

	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

In [11]:

```
cancer_df.objetivo.value_counts(True)
```

Out[11]:

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

Implementación

Predicción de los datos mediante regresión lineal

Vamos a intentar predecir resultados en función al dataset anteriormente cargado mediante regresión lineal, donde demostraremos que los resultados no se encuentran entre los posibles valores *0 y 1*

In [12]:

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
```

In [13]:

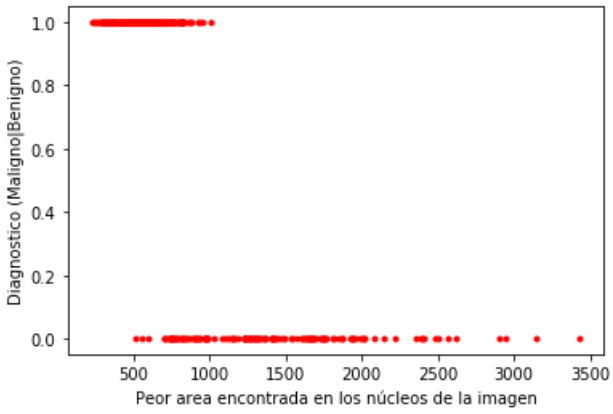
```
train_df, test_df = train_test_split(cancer_df, test_size=0.4)

variables_entrenamiento = cancer_datos["feature_names"]
variables_objetivo = "objetivo"
```

In [14]:

```
columna_entrenamiento = "worst area"

plt.plot(train_df[columna_entrenamiento], train_df.objetivo, '.r')
plt.xlabel("Peor area encontrada en los núcleos de la imagen")
plt.ylabel("Diagnostico (Maligno|Benigno)")
```

Out[14]:

```
Text(0,0.5,'Diagnostico (Maligno|Benigno)')
```



In [15]:

```
modelo_ols = LinearRegression()

modelo_ols.fit(train_df[[columna_entrenamiento]],
               train_df[variables_objetivo])

predicciones = modelo_ols.predict(test_df[[columna_entrenamiento]])

predicciones[:10]
```

Out[15]:

```
array([0.11885511, 0.15354393, 0.52291572, 0.12784851, 0.80826346,
       0.9522221 , 0.66558959, 0.88175239, 0.85033973, 0.57931719])
```

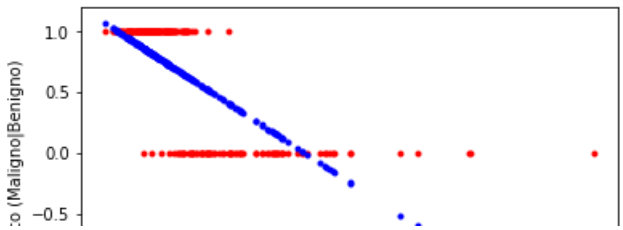
Como hemos observado en las predicciones de Regresión lineal, los resultados predecidos se encuentran más allá de los máximos y mínimos (0 y 1). Si obtenemos un gráfico de esta predicción, veremos más claramente como la linea sale de los margenes de los valores de las clases

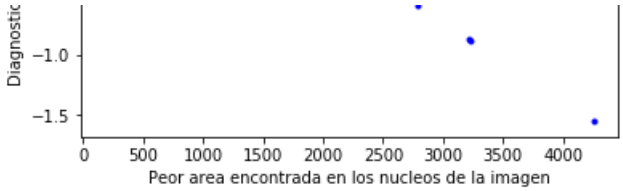
In [16]:

```
plt.plot(test_df[columna_entrenamiento], test_df.objetivo, '.r')
plt.plot(test_df[columna_entrenamiento], predicciones, '.b')
plt.xlabel("Peor area encontrada en los nucleos de la imagen")
plt.ylabel("Diagnostico (Maligno|Benigno)")
```

Out[16]:

```
Text(0,0.5,'Diagnostico (Maligno|Benigno)')
```





Aplicación de la función teórica

In [17]:

```
from ipywidgets import interact

def funcion_logistica(x, L=1, k=1, x0=0):
    return L / (1 + np.exp(-k*(x-x0)))


@interact(L=range(1,10), k=range(-5, 5), x0=range(0,10))
def plot_funcion_logit(L, k, x0):
    x = np.linspace(-5*k, 5*k, 500)
    y = funcion_logistica(x, k=k, L=L, x0=x0)
    plt.figure(1)
    plt.plot(x, y)
#    plt.show()
```

In [18]:

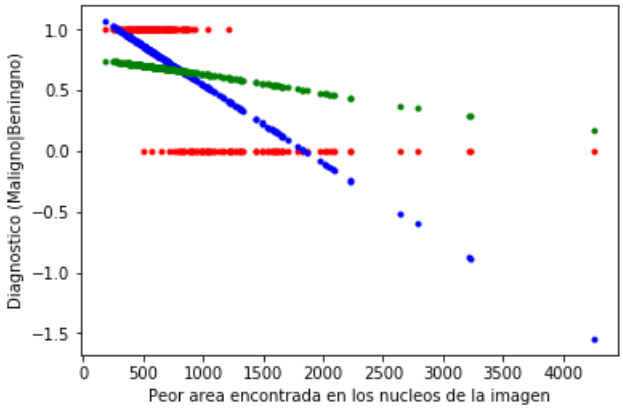
```
predicciones_probabilidades = list(map(funcion_logistica, predicciones))
```

In [19]:

```
plt.plot(test_df[columna_entrenamiento], test_df.objetivo, '.r')
plt.plot(test_df[columna_entrenamiento], predicciones, '.b')
plt.plot(test_df[columna_entrenamiento], predicciones_probabilidades, '.g')

plt.xlabel("Peor area encontrada en los nucleos de la imagen")
plt.ylabel("Diagnostico (Maligno|Beningno)")
```

Out[19]:

```
Text(0,0.5,'Diagnostico (Maligno|Beningno)')
```

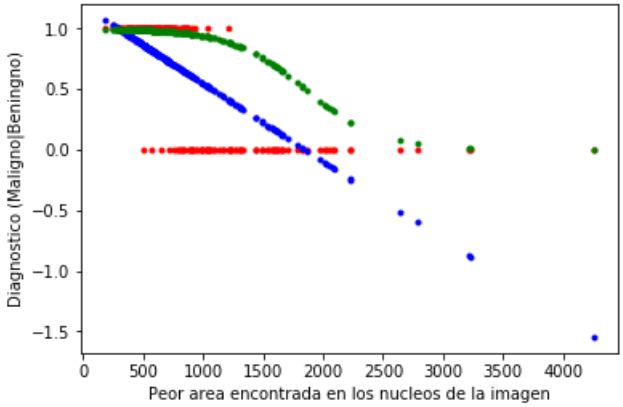


In [20]:

```
from functools import partial

funcion_logit_k5 = partial(funcion_logistica, k=5)

predicciones_probabilidades = list(map(funcion_logit_k5, predicciones))

plt.plot(test_df[columna_entrenamiento], test_df.objetivo, '.r')
plt.plot(test_df[columna_entrenamiento], predicciones, '.b')
plt.plot(test_df[columna_entrenamiento], predicciones_probabilidades, '.g')

plt.xlabel("Peor area encontrada en los nucleos de la imagen")
plt.ylabel("Diagnostico (Maligno|Beningno)")
```

Out[20]:

```
Text(0,0.5,'Diagnostico (Maligno|Beningno)')
```



Regresión Logística con Sklearn

In [21]:

```
from sklearn.linear_model import LogisticRegression
```

In [22]:

```
#Documentacion
LogisticRegression?
```

Cargamos las variables del dataframe y las ajustamos en datos de entrenamiento y datos de test

In [23]:

```
X = cancer_df[variables_entrenamiento]
y = cancer_df[variables_objetivo]
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.2)
```

Ajustamos el modelo y predecimos en función a la variable test de X. *clf* es un clasificador que por convención se llaman así al utilizarlos con ScikitLearn. Si observamos el resultado de las predicciones, vemos que ha predecido según la etiqueta objetivo (columna *objetivo* en el dataframe).

In [24]:

```
clf = LogisticRegression()
clf.fit(X_train, y_train)
predicciones = clf.predict(X_test)

predicciones[:10]
```

```
/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)
```

Out[24]:

```
array([0, 1, 0, 1, 1, 0, 0, 1, 1, 1])
```

El método *predict_proba* nos da más información de la columna objetivo.El método nos permite obtener la probabilidad de la predicción, ofreciendonos así un coeficiente de la posibilidad de que un cáncer sea maligno o benigno. Esto es muy útil en una aplicación para el mundo real, ya que con el método predict predecimos si el cáncer es maligno o benigno y con el método *predict_proba* podemos ver realmente las posibilidades de que sea maligno o benigno de cara a decirselo a un paciente (sin arriesgarnos a darle una respuesta absoluta y tener la posibilidad de fallar)

In [25]:

```
predicciones_probabilidades = clf.predict_proba(X_test)
predicciones_probabilidades[:10]
```

Out[25]:

```
array([[1.00000000e+00, 7.53628070e-16],
       [9.95463787e-03, 9.90045362e-01],
       [1.00000000e+00, 4.90386956e-11],
       [4.35955773e-02, 9.56404423e-01],
```

```
       [6.28861369e-02, 9.37113863e-01],
       [9.93008258e-01, 6.99174183e-03],
       [9.99945277e-01, 5.47228506e-05],
       [6.16577457e-03, 9.93834225e-01],
       [1.33981058e-03, 9.98660189e-01],
       [5.82927101e-03, 9.94170729e-01]])
```

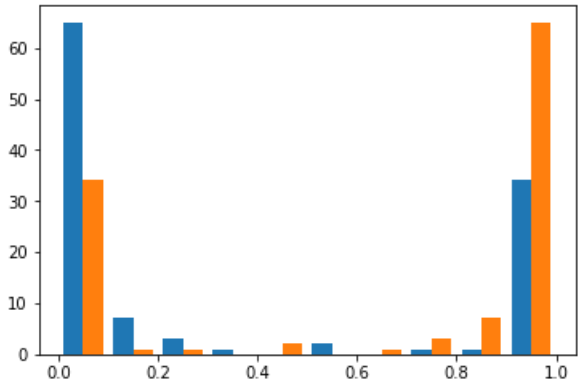
Obtenemos un histograma con las probabilidades de las predicciones realizadas

In [26]:

```
plt.hist(predicciones_probabilidades)
```

Out[26]:

```
([array([65.,  7.,  3.,  1.,  0.,  2.,  0.,  1.,  1., 34.]),
  array([34.,  1.,  1.,  0.,  2.,  0.,  1.,  3.,  7., 65.])],
 array([8.84843152e-17, 1.00000000e-01, 2.00000000e-01, 3.00000000e-01,
        4.00000000e-01, 5.00000000e-01, 6.00000000e-01, 7.00000000e-01,
        8.00000000e-01, 9.00000000e-01, 1.00000000e+00]),
 <a list of 2 Lists of Patches objects>)
```



Para exportar y visualizar mejor los resultados, exportamos las predicciones y la probabilidad a un dataframe, organizando las columnas del mismo de la siguiente manera:

- **objetivo**: valor real del dataframe original.
- **rediccion**: valor predecido por el algoritmo.
- 0: posibilidad de que sea cáncer maligno.
- 1: probabilidad de que sea cáncer benigno.

In [27]:

```
probs_df = pd.DataFrame(predicciones_probabilidades)
```

In [28]:

```
X = X_test.reset_index().copy()
X["objetivo"] = y_test.tolist()
X["prediccion"] = predicciones
X = pd.concat([X, probs_df], axis=1)
X[["objetivo","prediccion",0 ,1]].head(10)
```

Out[28]:

	objetivo	prediccion	0	1
0	0	0	1.000000	7.536281e-16
1	1	1	0.009955	9.900454e-01
2	0	0	1.000000	4.903870e-11
3	1	1	0.043596	9.564044e-01
4	1	1	0.062886	9.371139e-01
5	0	0	0.993008	6.991742e-03
6	0	0	0.999945	5.472285e-05
7	1	1	0.006166	9.938342e-01
8	1	1	0.001340	9.986602e-01

	objetivo	prediccion	0	1
8	1	1	0.001340	9.986602e-01
9	1	1	0.005829	9.941707e-01

In [29]:

```
probs_df.to_csv("prediccion_con_regresionLogistica.csv")
```