# Análisis de Datos y Aprendizaje Máquina con Tensorflow 2.0: Clasificación

2019/09/30

## 1 Naive Bayes

Objetivo: El aprendizaje máquina se puede usar para clasificación y regresión. Se comprenderá el concepto y funcionamiento de un clasificador, así como su evaluación y como implementarlo a un dataset.

- Referencia y documentación: https://scikit-learn.org/stable/modules/naive_bayes.html

Esta técnica está basada en el teorema de Bayes. Se utiliza para clasificar vectores de características. Se asume que las características son independientes dada la clase, es por esto la palabra 'naive'. Este clasificador trabaja bien incluso si las características son dependientes, además de que no sufre de sobreajuste.

*Nota: Se usará el mismo dataset con diferentes clasificadores para familiarizarse con los datos y poder hacer comparación (ventajas/desventajas) clara entre estos.*

$$P(y \mid x_1, \ldots, x_n) = \frac{P(y)P(x_1, \ldots x_n \mid y)}{P(x_1, \ldots, x_n)}$$

Usando la regla de al cadena

$$P(y \mid x_1, \ldots, x_n) = \frac{P(y)\prod_{i=1}^{n} P(x_i \mid y)}{P(x_1, \ldots, x_n)}$$

Dado que $P(x_1, \ldots, x_n)$ es constante dada la entrada, se puede usar la siguiente regla de clasificación:

$$P(y \mid x_1, \ldots, x_n) \propto P(y)\prod_{i=1}^{n} P(x_i \mid y)$$

$$\hat{y} = \arg\max_y P(y)\prod_{i=1}^{n} P(x_i \mid y)$$

Se encuentra la clase $y$ con máxima probabilidad.

- Bernoulli Naive Bayes implementa la regla de decisión

$$P(x_i|y) = P(i|y)x_i + (1 - P(i|y))(1 - x_i)$$

Se asume que las características son binarias, si no es el caso el parámetro 'binarize' debe ser indicado

- Complement Naive Bayes sirve para datos no balanceados y ha mostrado superar a Multinomial Naive Bayes para clasificación de texto. Este algoritmo calcula pesos.
- Gaussian Naive Bayes implementa el algoritmo Gaussian Naive Bayes para la clasificación. Se asume que la probabilidad de las características es gaussiana:

$$P(x_i \mid y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp(-\frac{(x_i - \mu_y)^2}{2\sigma_y^2})$$

Los parámetros $\sigma_y$ y $\mu_y$ de los valores $x$ están asociados con la clase $y$

```
In [1]: import sklearn
        import pandas as pd
        import seaborn as sns
        import matplotlib.pyplot as plt
```

## 1.1 Análisis exploratorio

### 1.1.1 Etiquetas de clase a valor numérico

- Diagnosis (M = malignant, B = benign)

```
In [2]: %matplotlib inline

        df = pd.read_csv("data-breast.csv",index_col=0)

        df.head(10)
```

Out[2]:

| id | diagnosis | radius_mean | texture_mean | perimeter_mean | area_mean | \ |
|---|---|---|---|---|---|---|
| 842302 | M | 17.99 | 10.38 | 122.80 | 1001.0 | |
| 842517 | M | 20.57 | 17.77 | 132.90 | 1326.0 | |
| 84300903 | M | 19.69 | 21.25 | 130.00 | 1203.0 | |
| 84348301 | M | 11.42 | 20.38 | 77.58 | 386.1 | |
| 84358402 | M | 20.29 | 14.34 | 135.10 | 1297.0 | |
| 843786 | M | 12.45 | 15.70 | 82.57 | 477.1 | |
| 844359 | M | 18.25 | 19.98 | 119.60 | 1040.0 | |
| 84458202 | M | 13.71 | 20.83 | 90.20 | 577.9 | |
| 844981 | M | 13.00 | 21.82 | 87.50 | 519.8 | |
| 84501001 | M | 12.46 | 24.04 | 83.97 | 475.9 | |

| id | smoothness_mean | compactness_mean | concavity_mean | \ |
|---|---|---|---|---|
| 842302 | 0.11840 | 0.27760 | 0.30010 | |
| 842517 | 0.08474 | 0.07864 | 0.08690 | |
| 84300903 | 0.10960 | 0.15990 | 0.19740 | |
| 84348301 | 0.14250 | 0.28390 | 0.24140 | |
| 84358402 | 0.10030 | 0.13280 | 0.19800 | |
| 843786 | 0.12780 | 0.17000 | 0.15780 | |
| 844359 | 0.09463 | 0.10900 | 0.11270 | |

```
84458202          0.11890           0.16450         0.09366
844981            0.12730           0.19320         0.18590
84501001          0.11860           0.23960         0.22730

          concave points_mean  symmetry_mean  ...  texture_worst  \
id                                            ...
842302                0.14710         0.2419  ...          17.33
842517                0.07017         0.1812  ...          23.41
84300903              0.12790         0.2069  ...          25.53
84348301              0.10520         0.2597  ...          26.50
84358402              0.10430         0.1809  ...          16.67
843786                0.08089         0.2087  ...          23.75
844359                0.07400         0.1794  ...          27.66
84458202              0.05985         0.2196  ...          28.14
844981                0.09353         0.2350  ...          30.73
84501001              0.08543         0.2030  ...          40.68

          perimeter_worst  area_worst  smoothness_worst  compactness_worst  \
id
842302             184.60      2019.0            0.1622             0.6656
842517             158.80      1956.0            0.1238             0.1866
84300903           152.50      1709.0            0.1444             0.4245
84348301            98.87       567.7            0.2098             0.8663
84358402           152.20      1575.0            0.1374             0.2050
843786             103.40       741.6            0.1791             0.5249
844359             153.20      1606.0            0.1442             0.2576
84458202           110.60       897.0            0.1654             0.3682
844981             106.20       739.3            0.1703             0.5401
84501001            97.65       711.4            0.1853             1.0580

          concavity_worst  concave points_worst  symmetry_worst  \
id
842302             0.7119                0.2654          0.4601
842517             0.2416                0.1860          0.2750
84300903           0.4504                0.2430          0.3613
84348301           0.6869                0.2575          0.6638
84358402           0.4000                0.1625          0.2364
843786             0.5355                0.1741          0.3985
844359             0.3784                0.1932          0.3063
84458202           0.2678                0.1556          0.3196
844981             0.5390                0.2060          0.4378
84501001           1.1050                0.2210          0.4366

          fractal_dimension_worst  Unnamed: 32
id
842302                    0.11890          NaN
842517                    0.08902          NaN
84300903                  0.08758          NaN
84348301                  0.17300          NaN
```

```
84358402                  0.07678          NaN
843786                    0.12440          NaN
844359                    0.08368          NaN
84458202                  0.11510          NaN
844981                    0.10720          NaN
84501001                  0.20750          NaN

[10 rows x 32 columns]
```

In [3]:
```
df.iloc[:,1:].describe()
```

Out[3]:

| | radius_mean | texture_mean | perimeter_mean | area_mean |
|---|---|---|---|---|
| count | 569.000000 | 569.000000 | 569.000000 | 569.000000 |
| mean | 14.127292 | 19.289649 | 91.969033 | 654.889104 |
| std | 3.524049 | 4.301036 | 24.298981 | 351.914129 |
| min | 6.981000 | 9.710000 | 43.790000 | 143.500000 |
| 25% | 11.700000 | 16.170000 | 75.170000 | 420.300000 |
| 50% | 13.370000 | 18.840000 | 86.240000 | 551.100000 |
| 75% | 15.780000 | 21.800000 | 104.100000 | 782.700000 |
| max | 28.110000 | 39.280000 | 188.500000 | 2501.000000 |

| | smoothness_mean | compactness_mean | concavity_mean | concave points_mean |
|---|---|---|---|---|
| count | 569.000000 | 569.000000 | 569.000000 | 569.000000 |
| mean | 0.096360 | 0.104341 | 0.088799 | 0.048919 |
| std | 0.014064 | 0.052813 | 0.079720 | 0.038803 |
| min | 0.052630 | 0.019380 | 0.000000 | 0.000000 |
| 25% | 0.086370 | 0.064920 | 0.029560 | 0.020310 |
| 50% | 0.095870 | 0.092630 | 0.061540 | 0.033500 |
| 75% | 0.105300 | 0.130400 | 0.130700 | 0.074000 |
| max | 0.163400 | 0.345400 | 0.426800 | 0.201200 |

| | symmetry_mean | fractal_dimension_mean | ... | texture_worst |
|---|---|---|---|---|
| count | 569.000000 | 569.000000 | ... | 569.000000 |
| mean | 0.181162 | 0.062798 | ... | 25.677223 |
| std | 0.027414 | 0.007060 | ... | 6.146258 |
| min | 0.106000 | 0.049960 | ... | 12.020000 |
| 25% | 0.161900 | 0.057700 | ... | 21.080000 |
| 50% | 0.179200 | 0.061540 | ... | 25.410000 |
| 75% | 0.195700 | 0.066120 | ... | 29.720000 |
| max | 0.304000 | 0.097440 | ... | 49.540000 |

| | perimeter_worst | area_worst | smoothness_worst | compactness_worst |
|---|---|---|---|---|
| count | 569.000000 | 569.000000 | 569.000000 | 569.000000 |
| mean | 107.261213 | 880.583128 | 0.132369 | 0.254265 |
| std | 33.602542 | 569.356993 | 0.022832 | 0.157336 |
| min | 50.410000 | 185.200000 | 0.071170 | 0.027290 |
| 25% | 84.110000 | 515.300000 | 0.116600 | 0.147200 |
| 50% | 97.660000 | 686.500000 | 0.131300 | 0.211900 |
| 75% | 125.400000 | 1084.000000 | 0.146000 | 0.339100 |
| max | 251.200000 | 4254.000000 | 0.222600 | 1.058000 |

```
       concavity_worst  concave points_worst  symmetry_worst  \
count       569.000000            569.000000      569.000000
mean          0.272188              0.114606        0.290076
std           0.208624              0.065732        0.061867
min           0.000000              0.000000        0.156500
25%           0.114500              0.064930        0.250400
50%           0.226700              0.099930        0.282200
75%           0.382900              0.161400        0.317900
max           1.252000              0.291000        0.663800

       fractal_dimension_worst  Unnamed: 32
count               569.000000          0.0
mean                  0.083946          NaN
std                   0.018061          NaN
min                   0.055040          NaN
25%                   0.071460          NaN
50%                   0.080040          NaN
75%                   0.092080          NaN
max                   0.207500          NaN

[8 rows x 31 columns]
```

In [4]:
```
df = df.replace({'B':0, 'M':1})
df
```

Out[4]:
```
          diagnosis  radius_mean  texture_mean  perimeter_mean  area_mean  \
id
842302            1        17.99         10.38          122.80     1001.0
842517            1        20.57         17.77          132.90     1326.0
84300903          1        19.69         21.25          130.00     1203.0
84348301          1        11.42         20.38           77.58      386.1
84358402          1        20.29         14.34          135.10     1297.0
...             ...          ...           ...             ...        ...
926424            1        21.56         22.39          142.00     1479.0
926682            1        20.13         28.25          131.20     1261.0
926954            1        16.60         28.08          108.30      858.1
927241            1        20.60         29.33          140.10     1265.0
92751             0         7.76         24.54           47.92      181.0

          smoothness_mean  compactness_mean  concavity_mean  \
id
842302            0.11840           0.27760         0.30010
842517            0.08474           0.07864         0.08690
84300903          0.10960           0.15990         0.19740
84348301          0.14250           0.28390         0.24140
84358402          0.10030           0.13280         0.19800
...                   ...               ...             ...
926424            0.11100           0.11590         0.24390
926682            0.09780           0.10340         0.14400
```

```
926954           0.08455          0.10230         0.09251
927241           0.11780          0.27700         0.35140
92751            0.05263          0.04362         0.00000

         concave points_mean  symmetry_mean  ...  texture_worst  \
id                                           ...
842302               0.14710         0.2419  ...          17.33
842517               0.07017         0.1812  ...          23.41
84300903             0.12790         0.2069  ...          25.53
84348301             0.10520         0.2597  ...          26.50
84358402             0.10430         0.1809  ...          16.67
...                      ...            ...  ...            ...
926424               0.13890         0.1726  ...          26.40
926682               0.09791         0.1752  ...          38.25
926954               0.05302         0.1590  ...          34.12
927241               0.15200         0.2397  ...          39.42
92751                0.00000         0.1587  ...          30.37

         perimeter_worst  area_worst  smoothness_worst  compactness_worst  \
id
842302             184.60      2019.0           0.16220            0.66560
842517             158.80      1956.0           0.12380            0.18660
84300903           152.50      1709.0           0.14440            0.42450
84348301            98.87       567.7           0.20980            0.86630
84358402           152.20      1575.0           0.13740            0.20500
...                   ...         ...               ...                ...
926424             166.10      2027.0           0.14100            0.21130
926682             155.00      1731.0           0.11660            0.19220
926954             126.70      1124.0           0.11390            0.30940
927241             184.60      1821.0           0.16500            0.86810
92751               59.16       268.6           0.08996            0.06444

         concavity_worst  concave points_worst  symmetry_worst  \
id
842302            0.7119                0.2654          0.4601
842517            0.2416                0.1860          0.2750
84300903          0.4504                0.2430          0.3613
84348301          0.6869                0.2575          0.6638
84358402          0.4000                0.1625          0.2364
...                  ...                   ...             ...
926424            0.4107                0.2216          0.2060
926682            0.3215                0.1628          0.2572
926954            0.3403                0.1418          0.2218
927241            0.9387                0.2650          0.4087
92751             0.0000                0.0000          0.2871

         fractal_dimension_worst  Unnamed: 32
id
842302                   0.11890          NaN
```

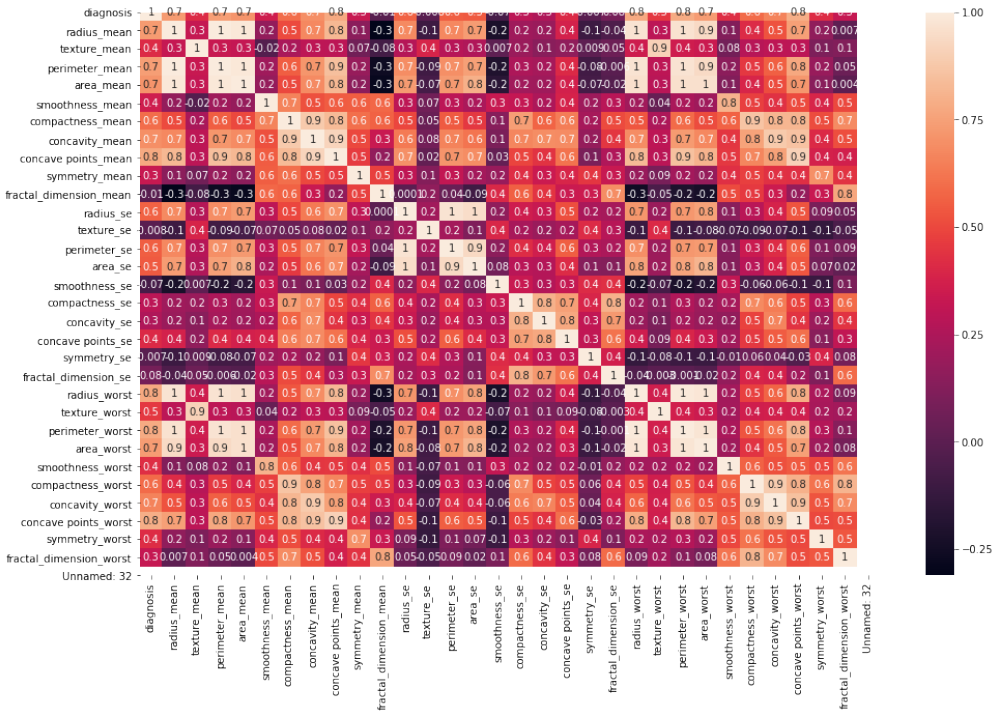
```
842517                  0.08902          NaN
84300903                0.08758          NaN
84348301                0.17300          NaN
84358402                0.07678          NaN
...                         ...          ...
926424                  0.07115          NaN
926682                  0.06637          NaN
926954                  0.07820          NaN
927241                  0.12400          NaN
92751                   0.07039          NaN

[569 rows x 32 columns]
```

## 1.2 Coeficientes de correlación

- Para selección de características se puede conocer que variables estan mas relacionadas con la variable de clase con la matriz de correlación
- El resultado del método 'corr()' de pandas se puede plotear con 'heatmap'

```
In [5]: import numpy as np
        corr = df.corr()
        plt.figure(figsize=(16, 10))
        sns.heatmap(corr, annot=True, fmt='.1g')
```

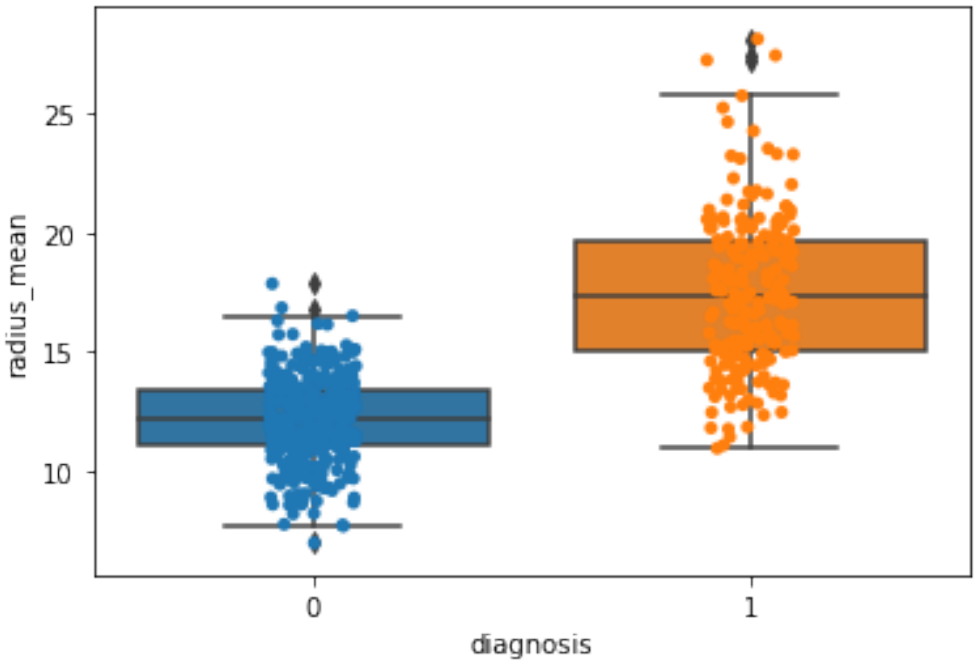
```
Out[5]: <matplotlib.axes._subplots.AxesSubplot at 0x7f345d871f10>
```

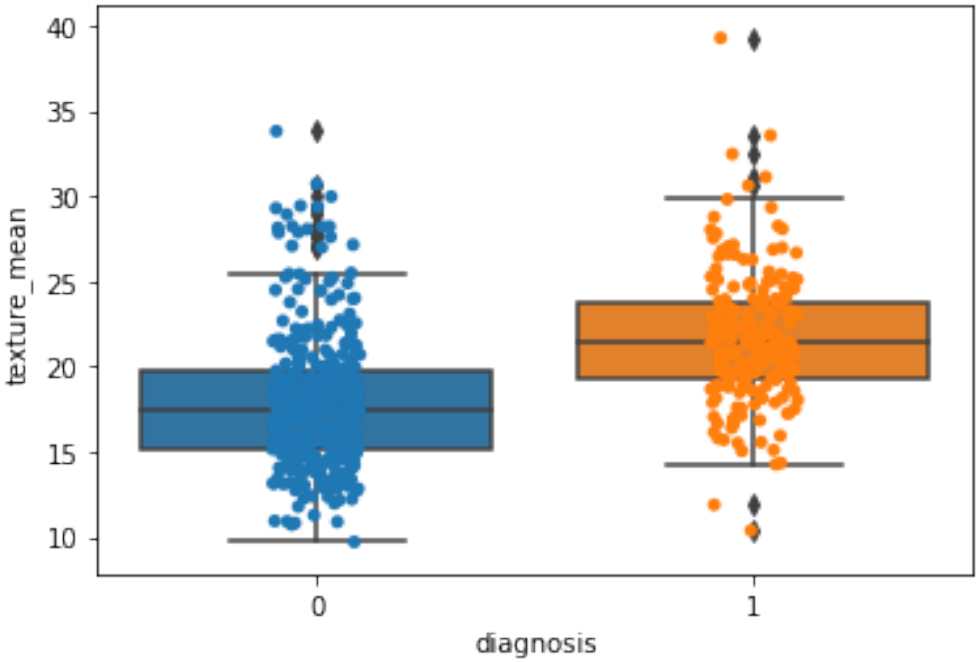


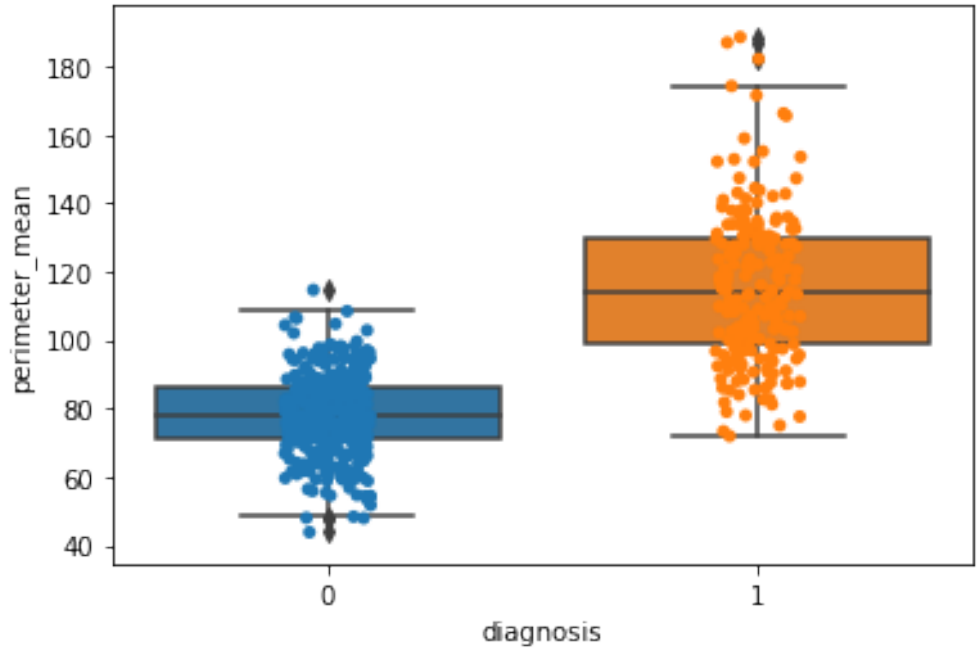
## 1.3 Boxplots de variables por clase

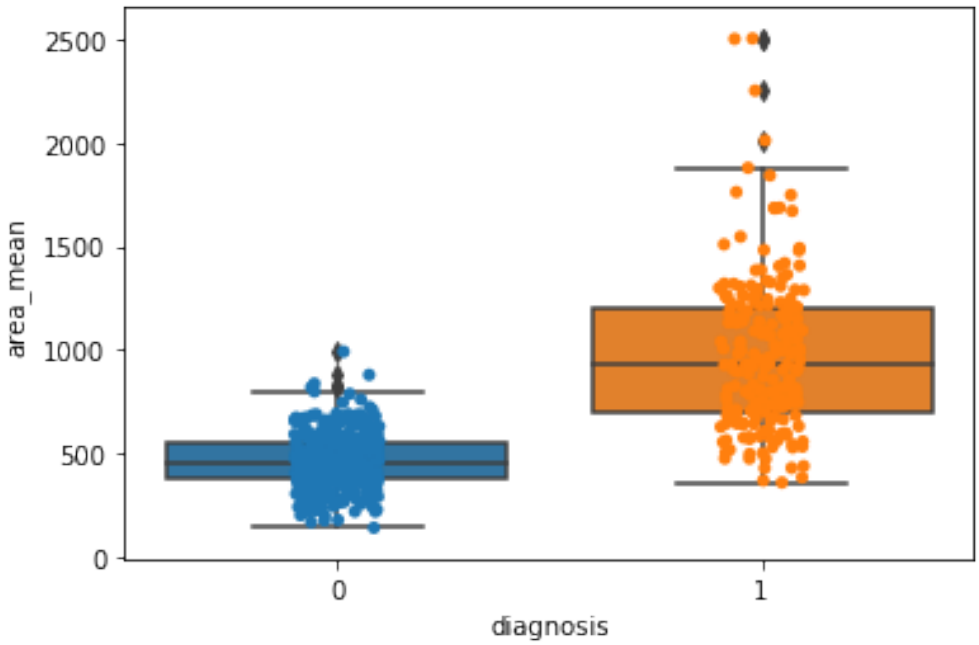
- 'sns.boxplot' recibe el nombre de la variable a plotear

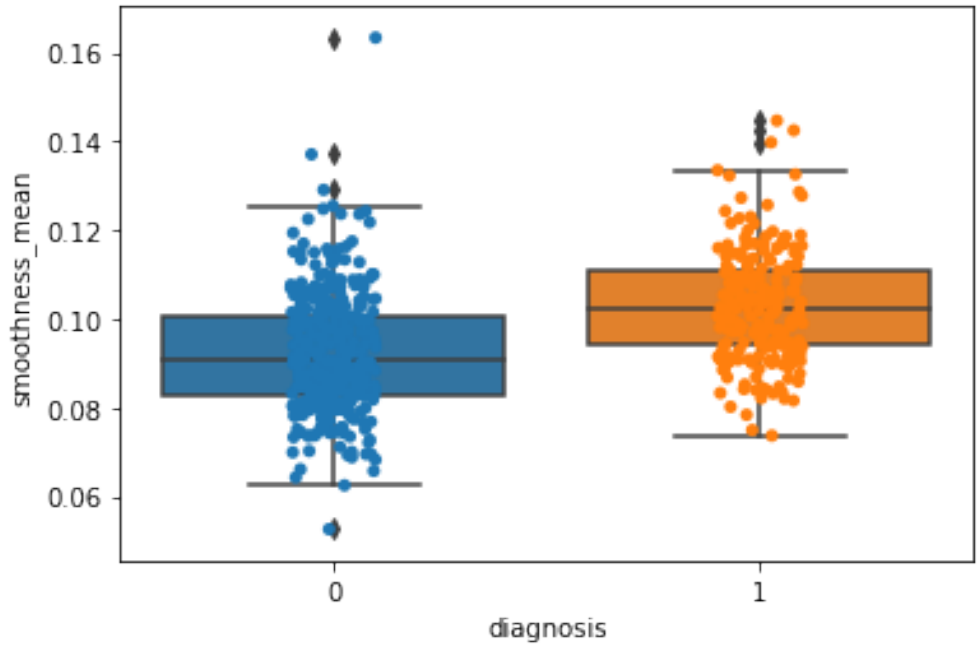
```
In [6]: import matplotlib as mpl
        mpl.rc('figure', max_open_warning = 0)
        l=list(df.columns)
        l[0:len(l)-1]
        for i in range(1,len(l)-1):
            sns.boxplot(x='diagnosis' ,y=l[i], data=df)
            sns.stripplot(x='diagnosis' ,y=l[i], data=df)
            plt.figure()
```

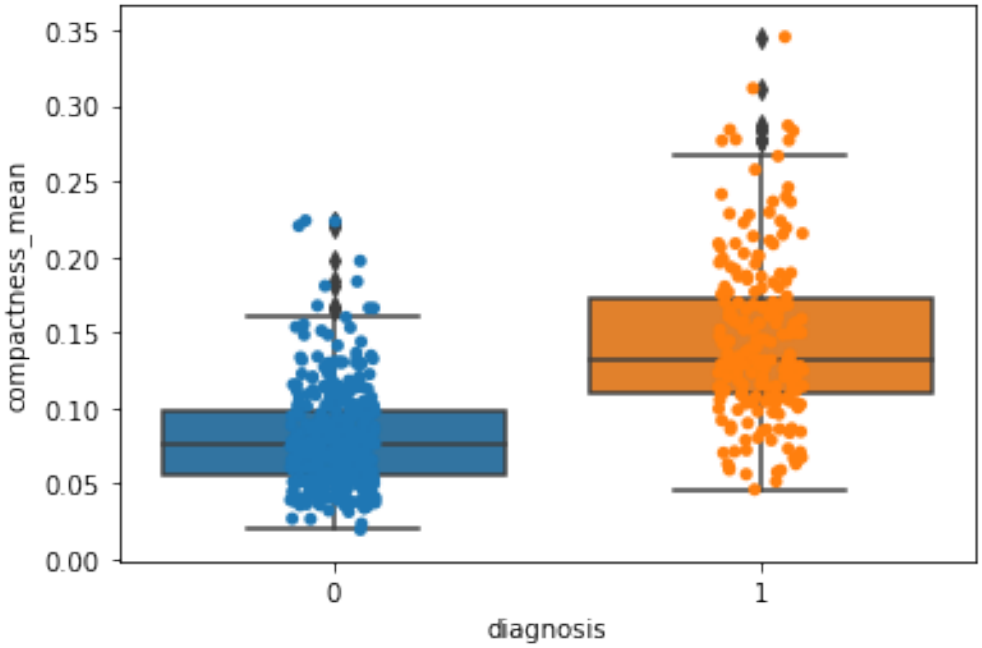


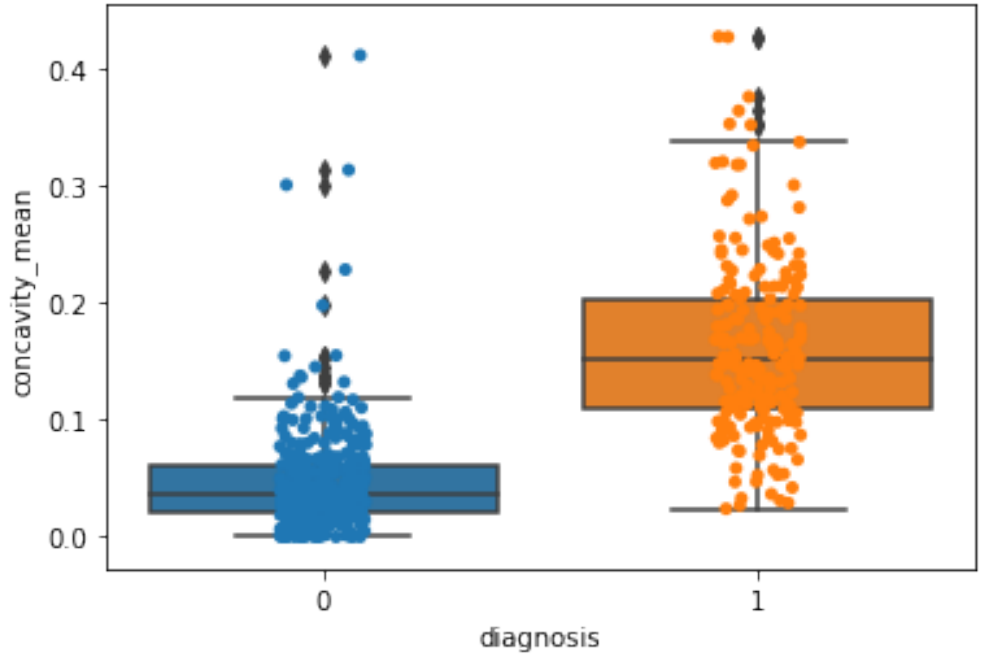

texture_mean
40
35
30
25
20
15
10
0
1
diagnosis

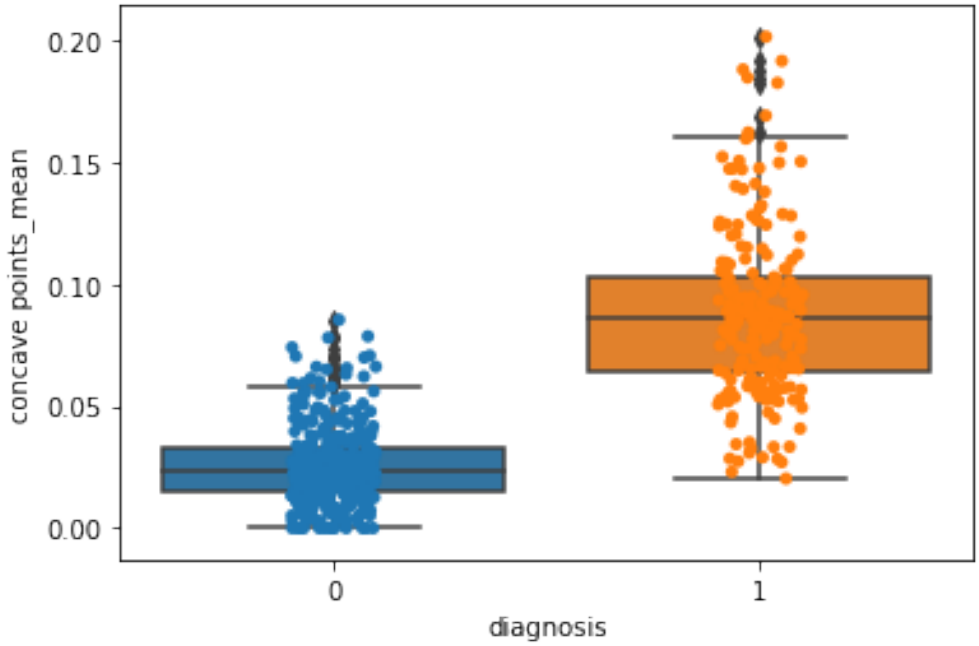

perimeter_mean
180
160
140
120
100
80
60
40
0
1
diagnosis

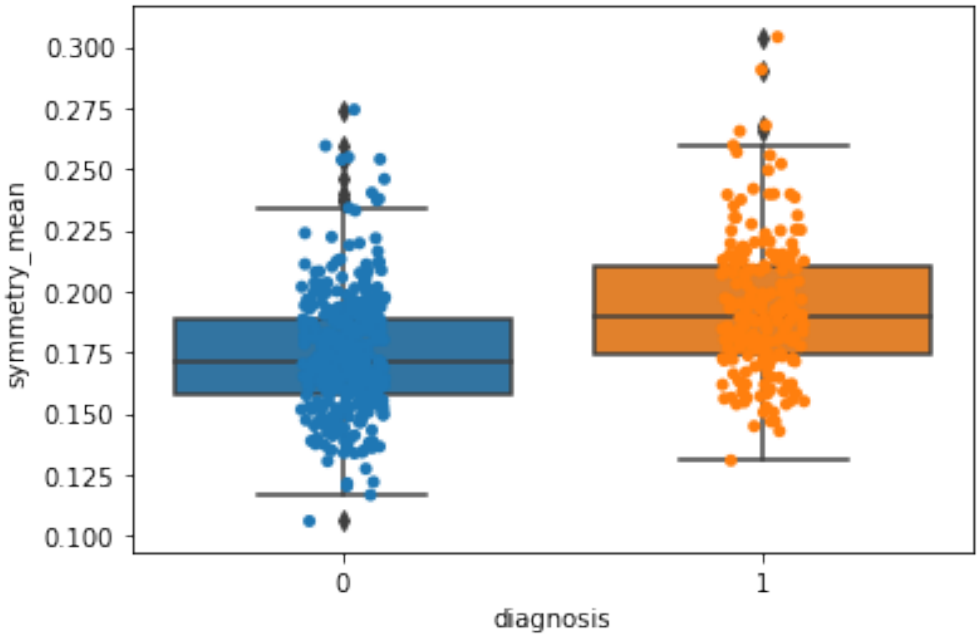

2500
2000
1500
1000
500
0
area_mean
0
1
diagnosis

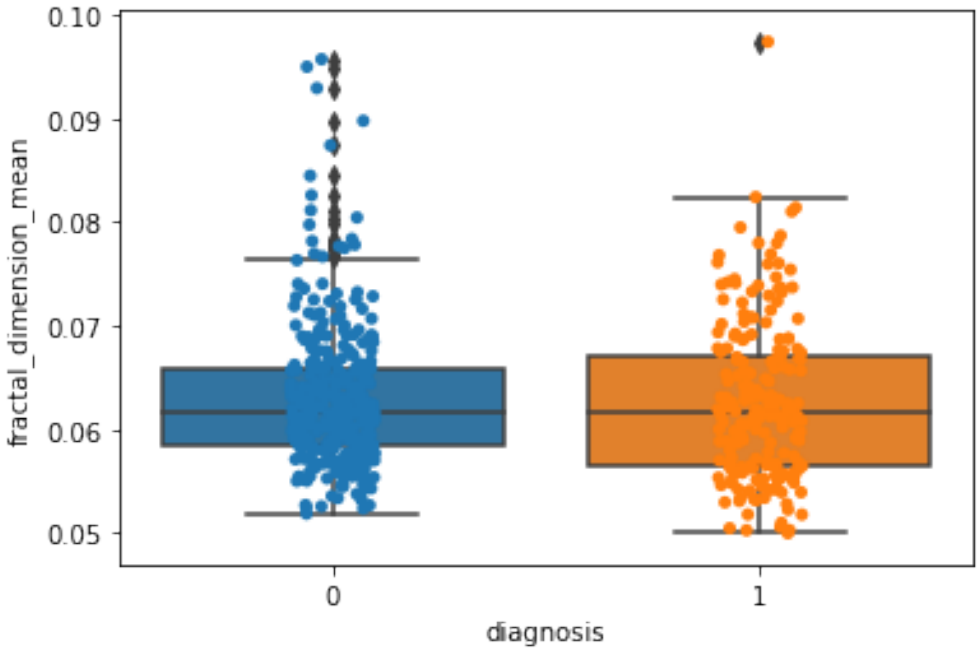

0.16
0.14
0.12
0.10
0.08
0.06
smoothness_mean
0
1
diagnosis

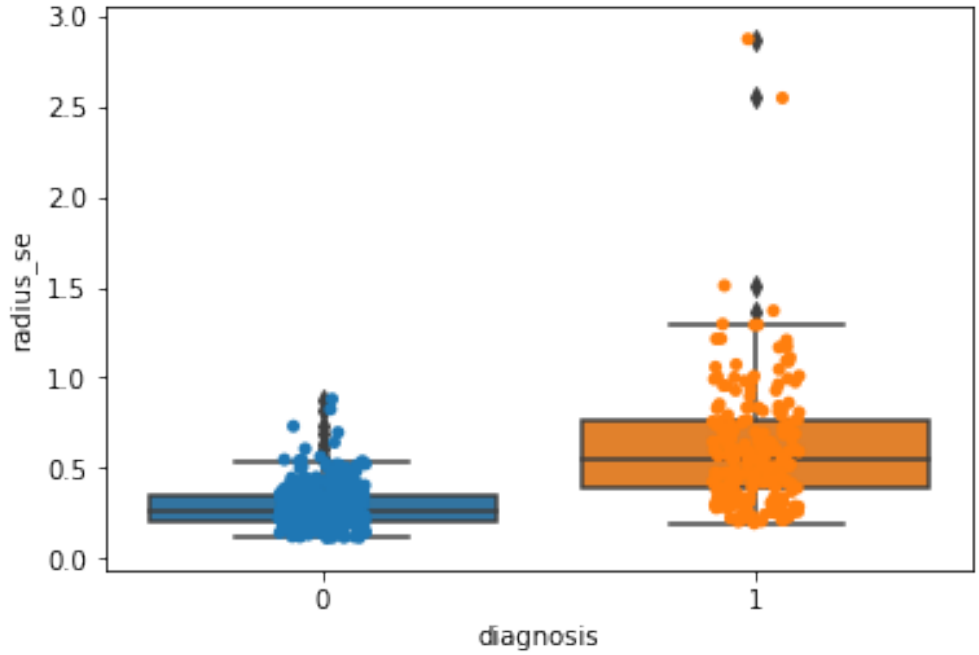

compactness_mean
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
0
1
diagnosis

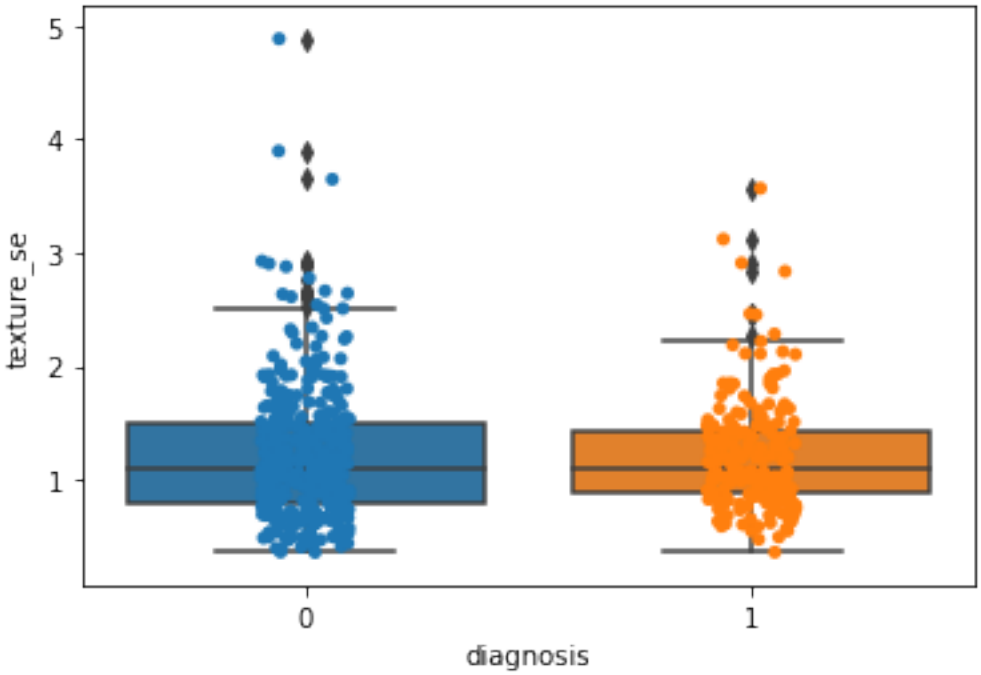

concavity_mean
0.4
0.3
0.2
0.1
0.0
0
1
diagnosis

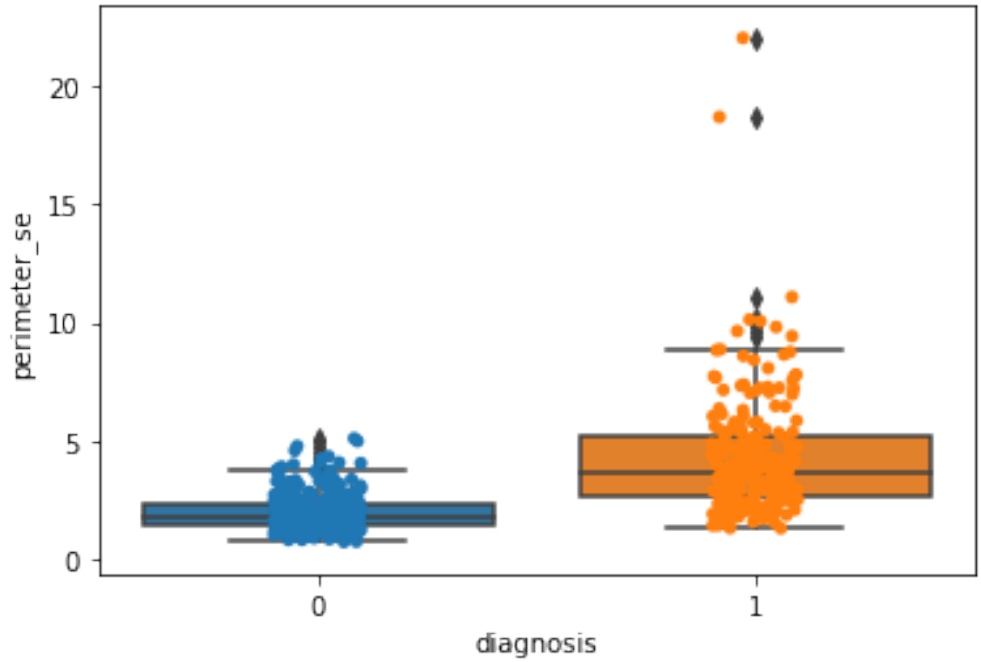

concave points_mean
0.20
0.15
0.10
0.05
0.00
0
1
diagnosis

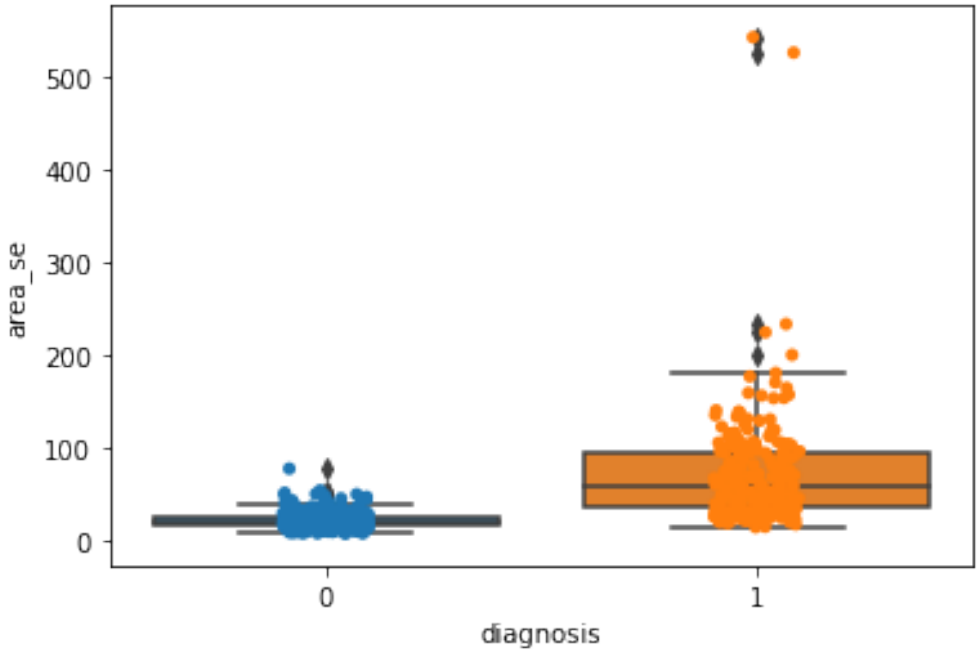

symmetry_mean
0.300
0.275
0.250
0.225
0.200
0.175
0.150
0.125
0.100
0
1
diagnosis

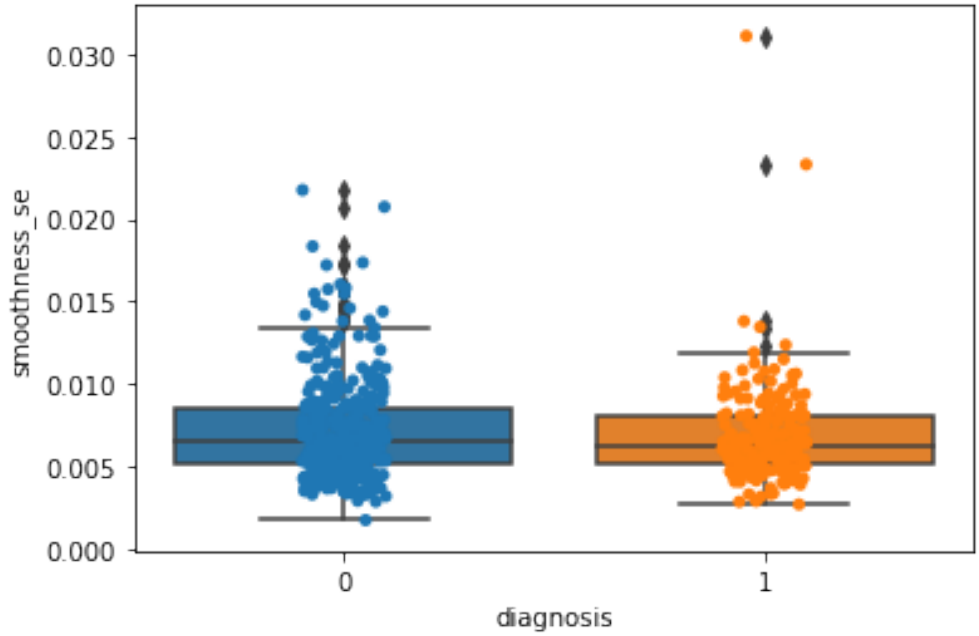

0.10
0.09
0.08
0.07
0.06
0.05
fractal_dimension_mean
0
1
diagnosis

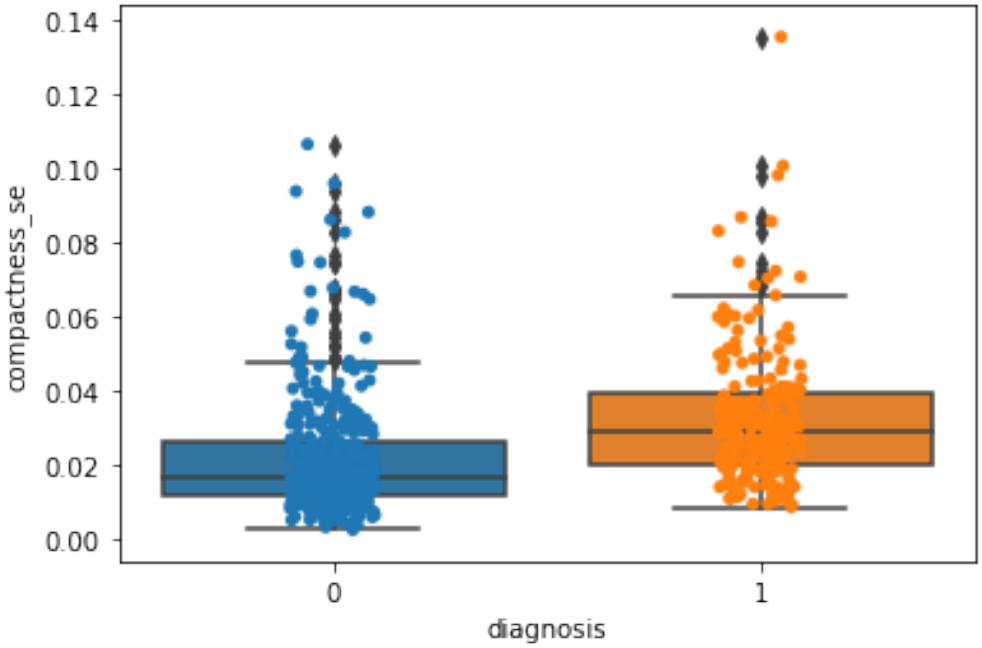

3.0
2.5
2.0
1.5
1.0
0.5
0.0
radius_se
0
1
diagnosis

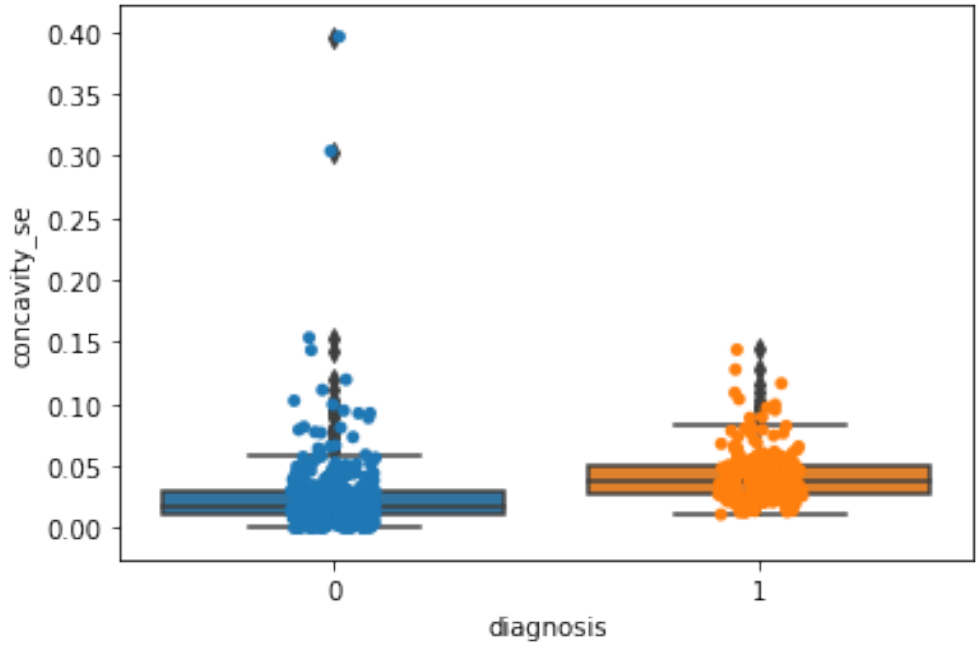

texture_se
5
4
3
2
1
0
1
diagnosis

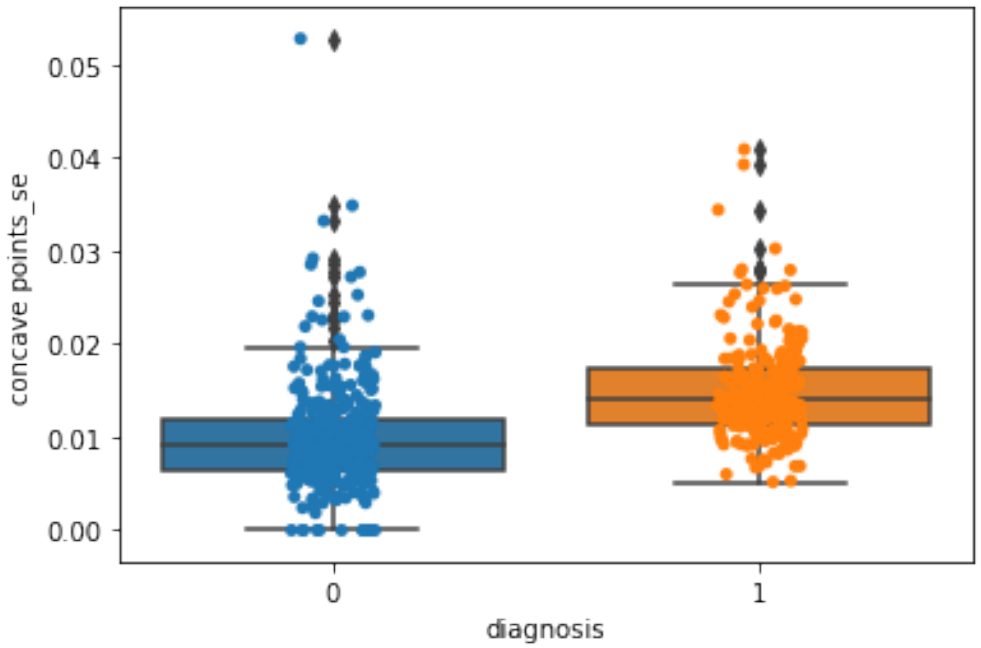

perimeter_se
20
15
10
5
0
0
1
diagnosis

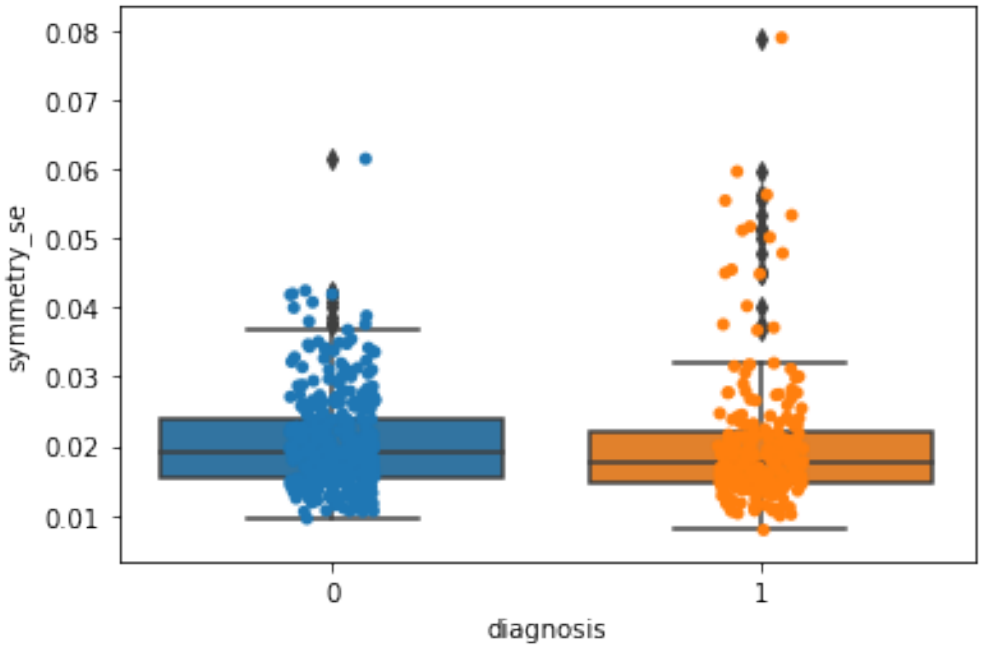

area_se
500
400
300
200
100
0
0
1
diagnosis

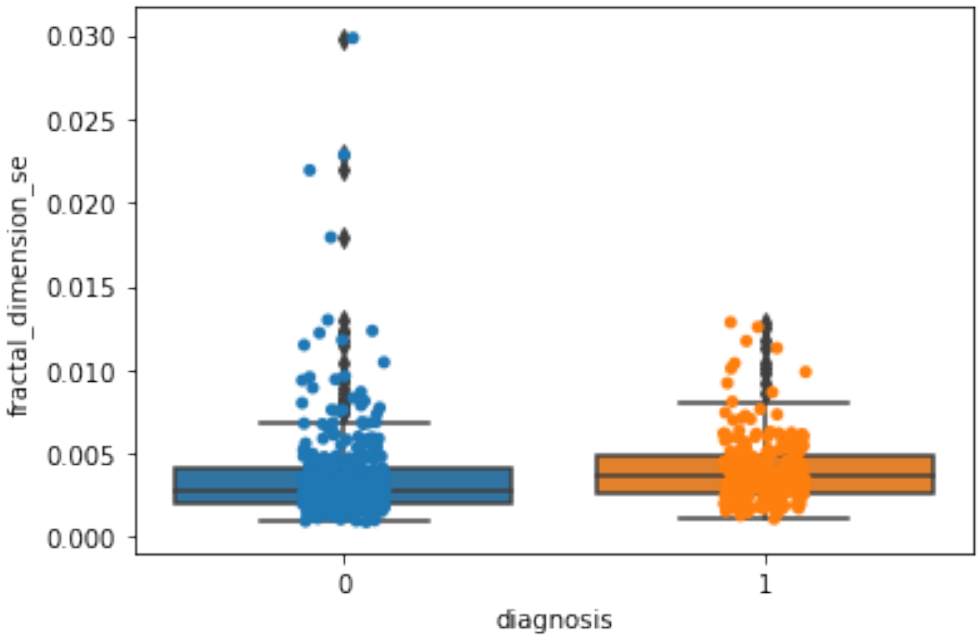

smoothness_se
0.030
0.025
0.020
0.015
0.010
0.005
0.000
0
1
diagnosis

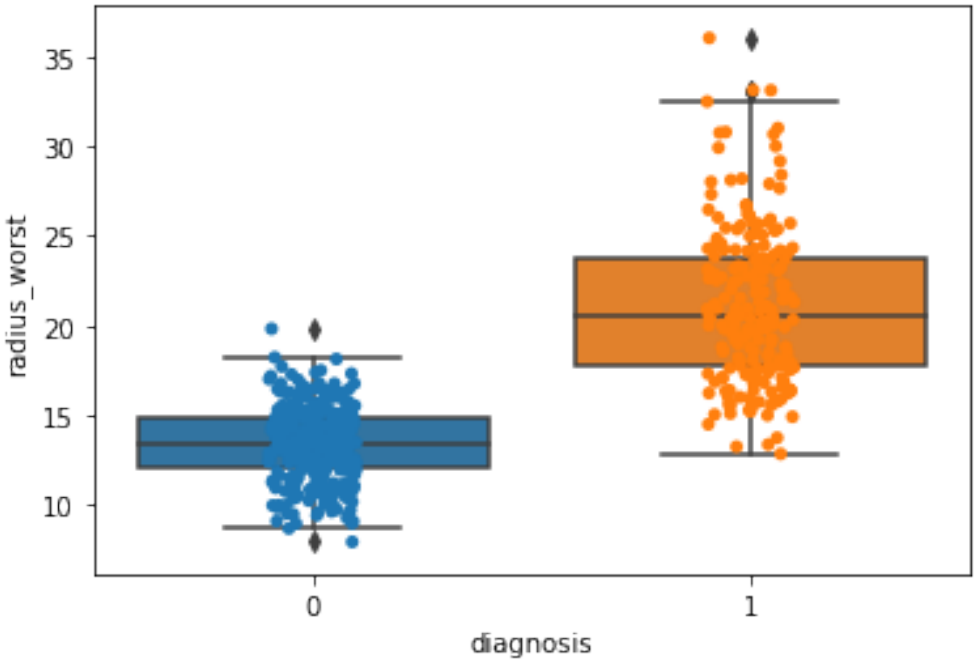

0.14
0.12
0.10
0.08
0.06
0.04
0.02
0.00
compactness_se
0
1
diagnosis

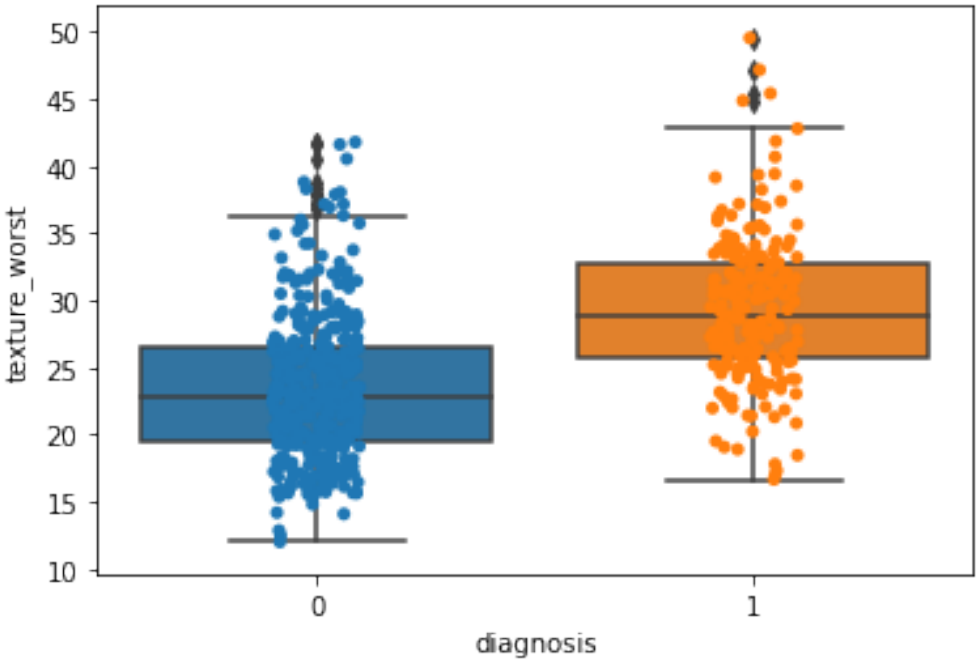

0.40
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
concavity_se
0
1
diagnosis

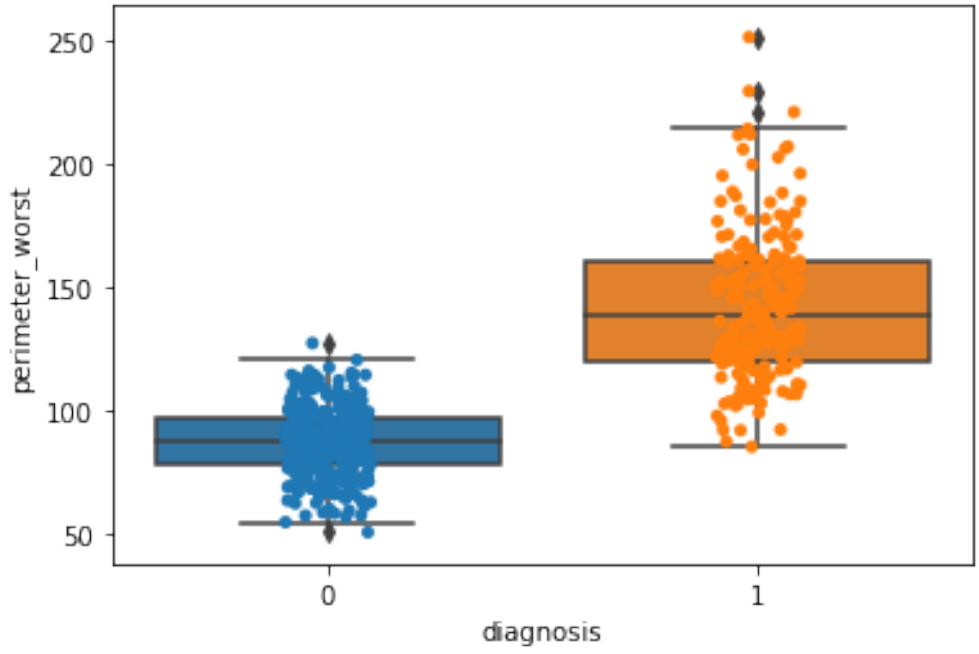

concave points_se
0.05
0.04
0.03
0.02
0.01
0.00
0
1
diagnosis

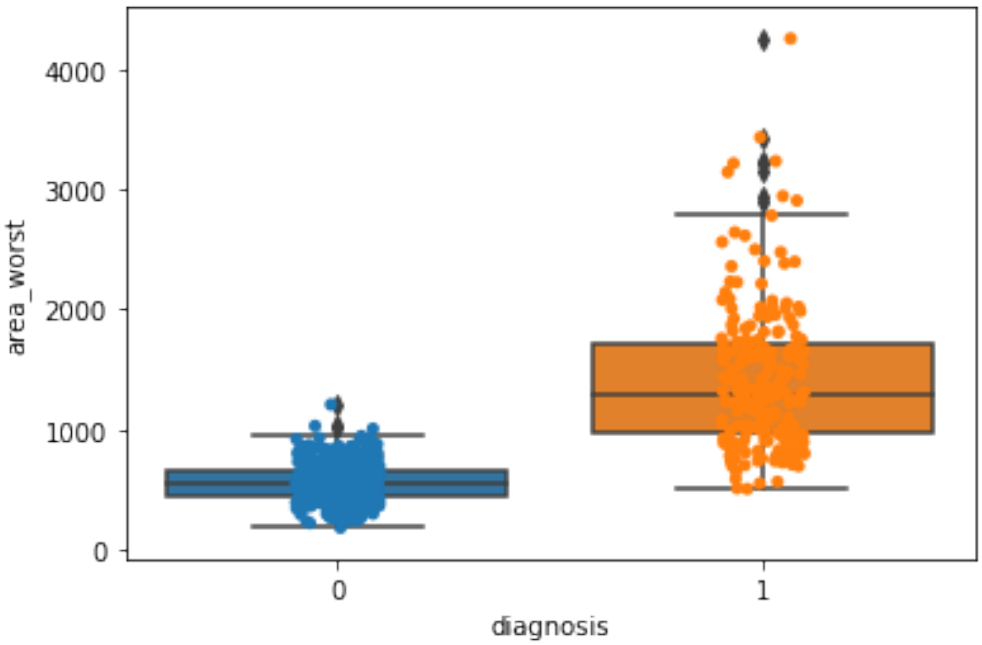

symmetry_se
0.08
0.07
0.06
0.05
0.04
0.03
0.02
0.01
0
1
diagnosis

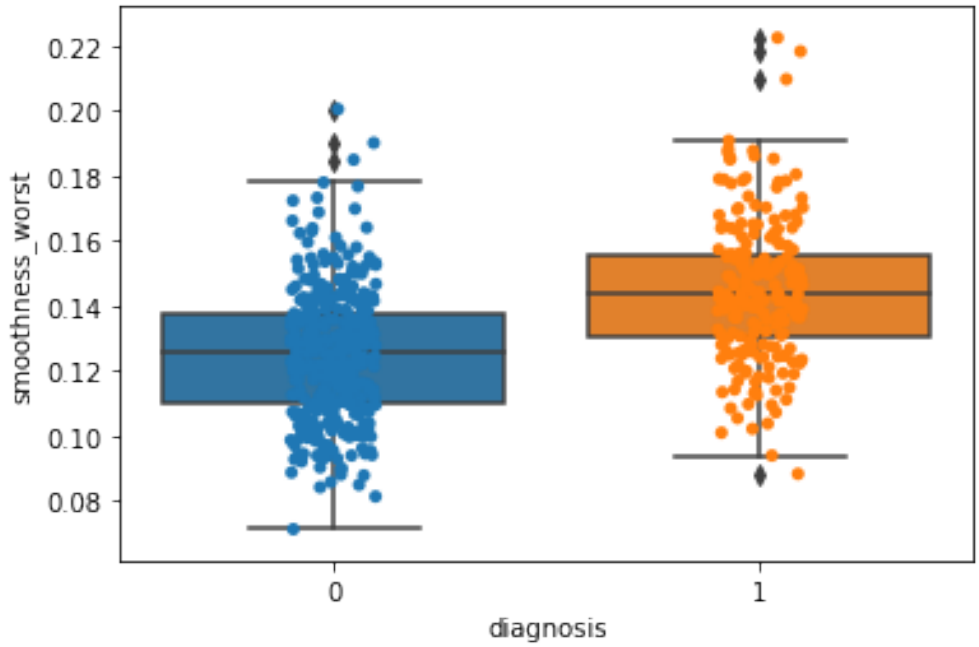

fractal_dimension_se
0.030
0.025
0.020
0.015
0.010
0.005
0.000
0
1
diagnosis

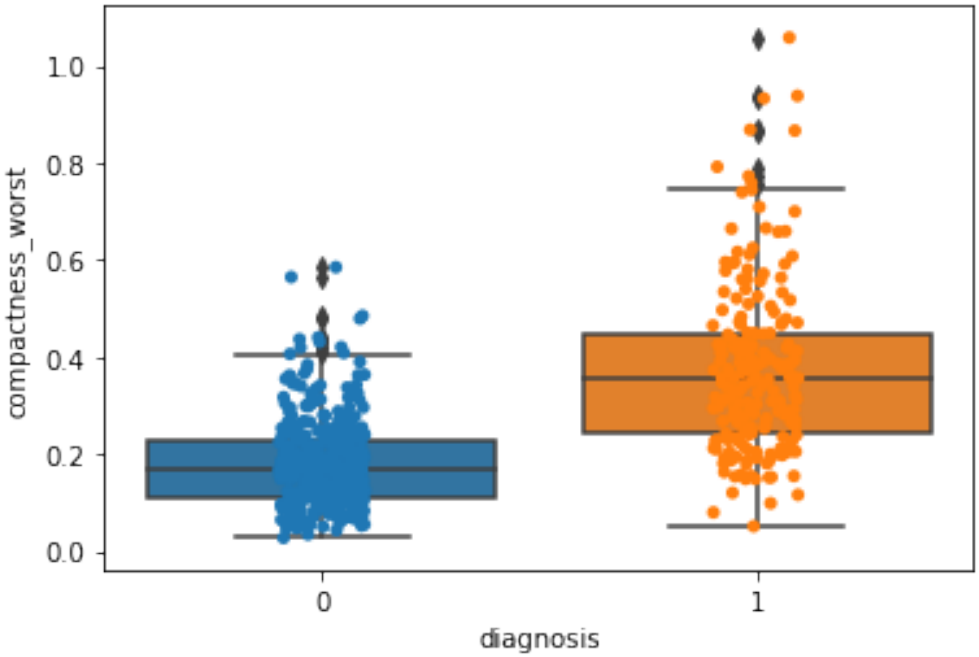

radius_worst
35
30
25
20
15
10
0
1
diagnosis

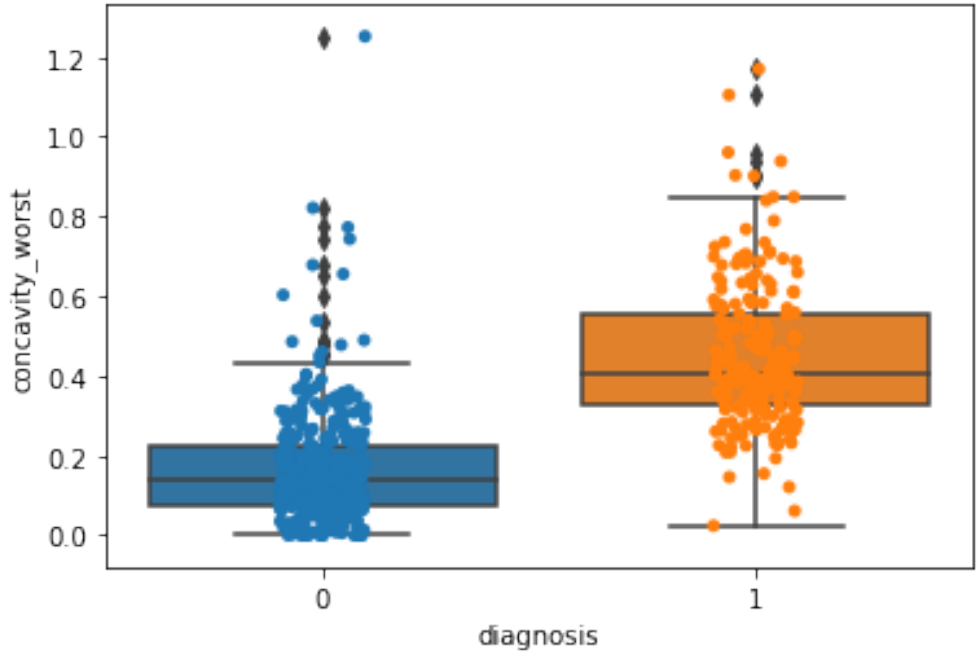

texture_worst
50
45
40
35
30
25
20
15
10
0
1
diagnosis

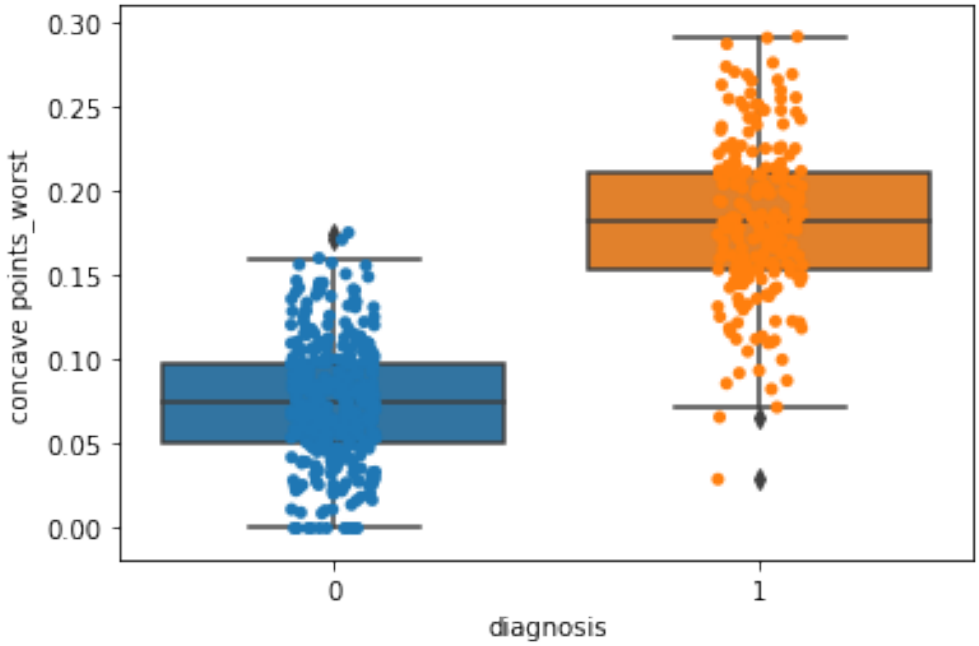

perimeter_worst
250
200
150
100
50
0
1
diagnosis

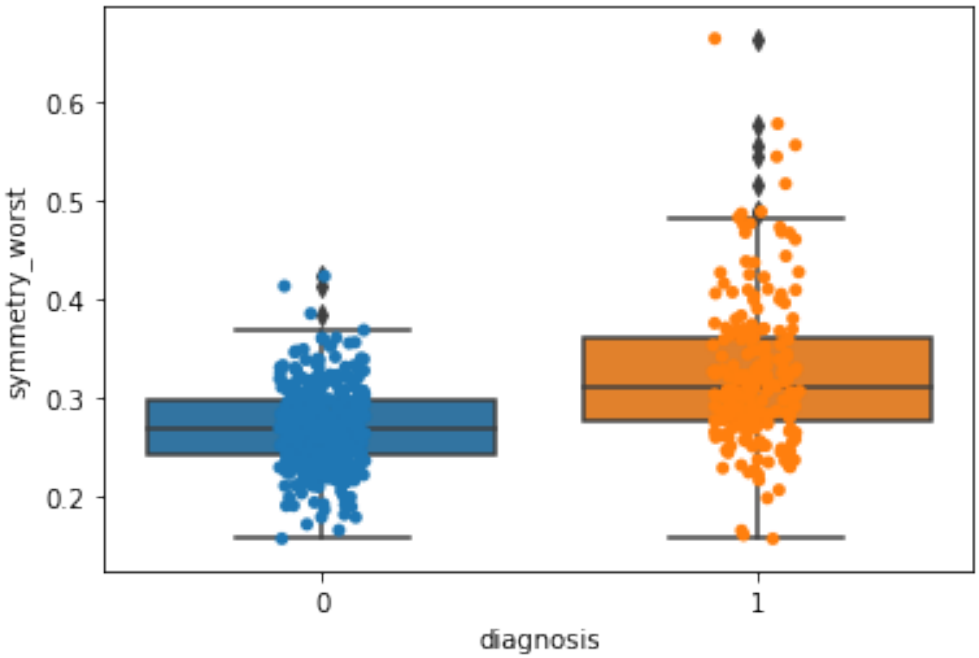

area_worst
4000
3000
2000
1000
0
0
1
diagnosis

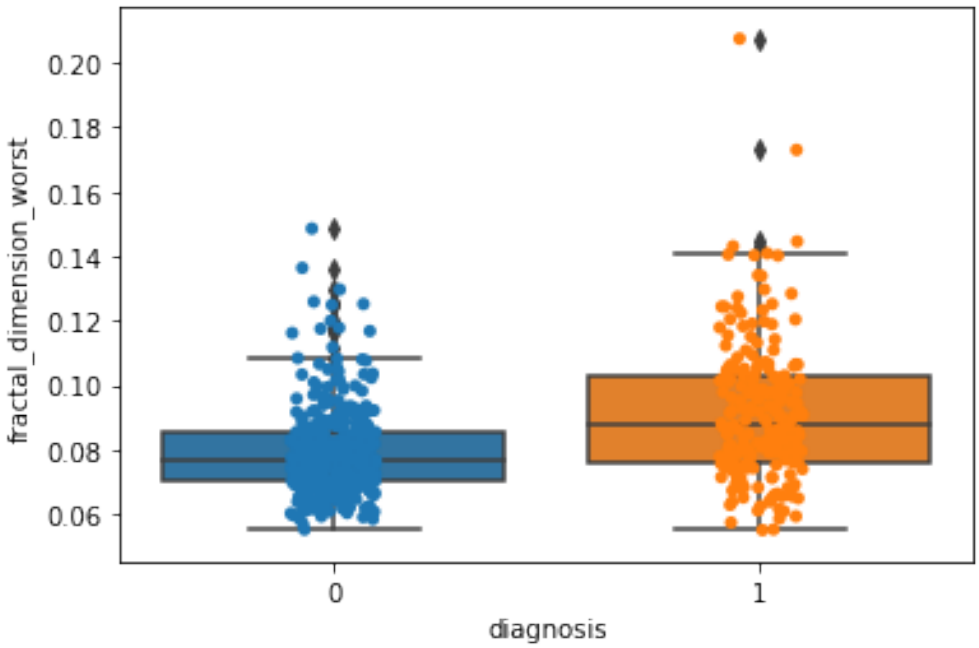

smoothness_worst
0.22
0.20
0.18
0.16
0.14
0.12
0.10
0.08
0
1
diagnosis


compactness_worst
1.0
0.8
0.6
0.4
0.2
0.0
0
1
diagnosis


concavity_worst
1.2
1.0
0.8
0.6
0.4
0.2
0.0
0
1
diagnosis


concave points_worst
0.30
0.25
0.20
0.15
0.10
0.05
0.00
0
1
diagnosis


symmetry_worst
0.6
0.5
0.4
0.3
0.2
0
1
diagnosis

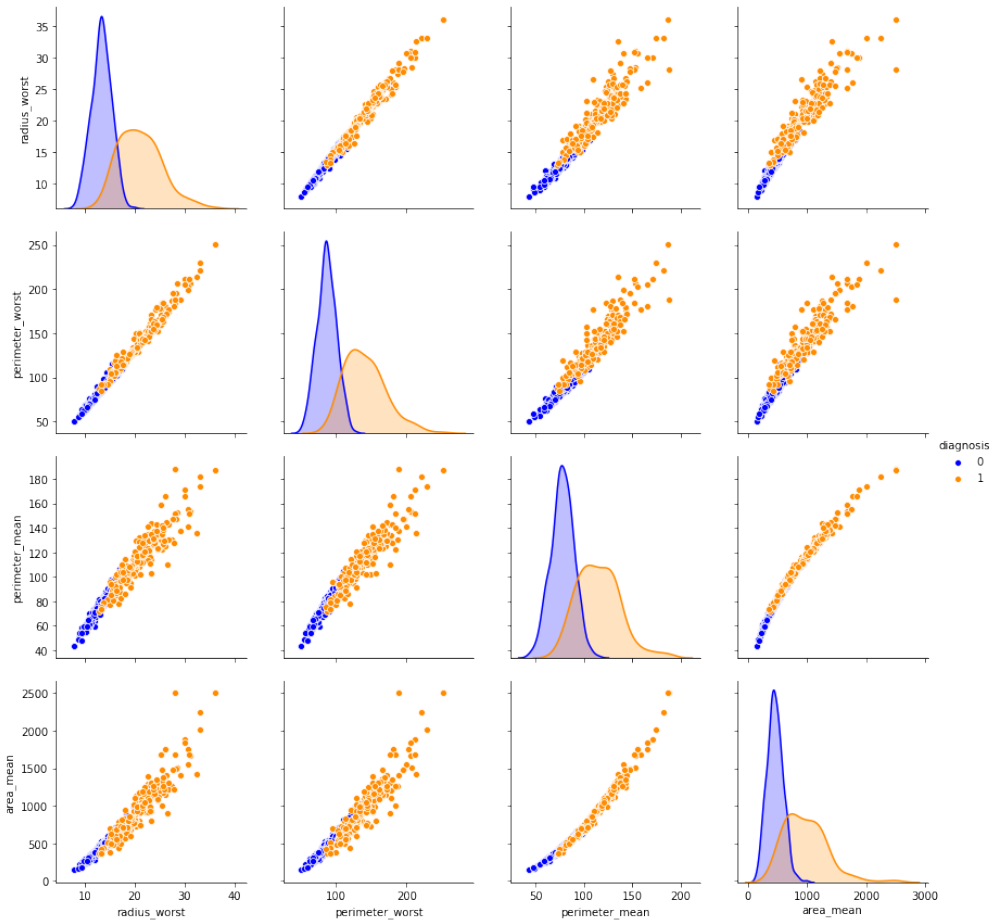



```
<Figure size 432x288 with 0 Axes>
```

## 1.4 Visualizar variables en plano 'x y'

- Con 'pairplot' se pueden visualizar las características de una forma clara y rápida. Se recibe como argumento la lista de variables a plotear

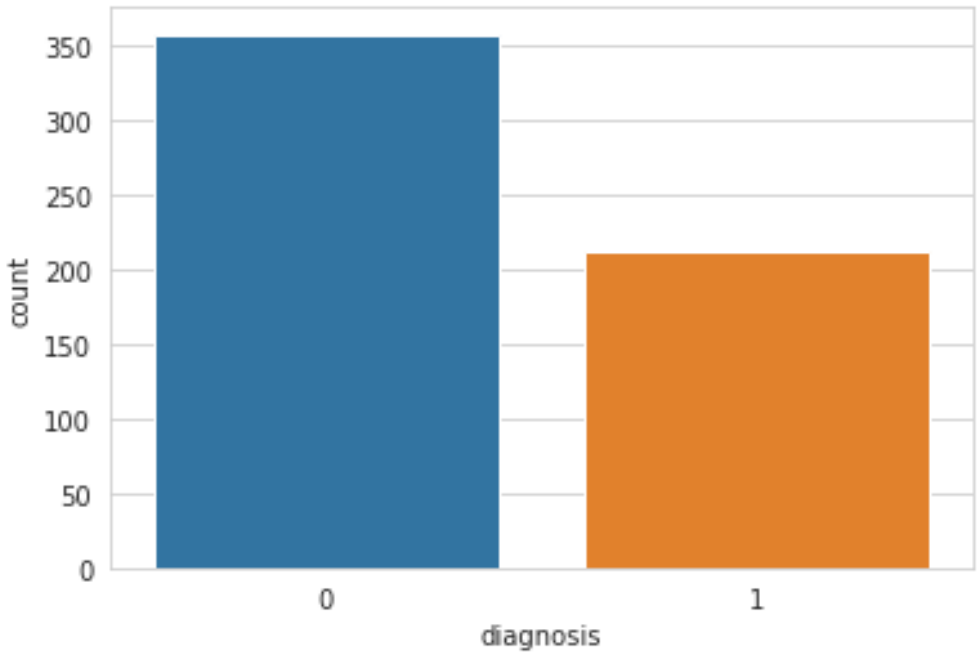
```
In [7]: sns.pairplot(df, vars=["radius_worst", "perimeter_worst", "perimeter_mean", "area_mean"],
                    palette=sns.color_palette(['blue', 'darkorange']), hue='diagnosis', height=3
        plt.show()
```




## 1.5 Conteo de clases

```
In [8]: sns.set_style('whitegrid')
        sns.countplot(x='diagnosis',data=df)
```

```
Out[8]: <matplotlib.axes._subplots.AxesSubplot at 0x7f345ab98c90>
```




### 1.6 Preparar datos para entrenamiento

```
In [9]: from sklearn.model_selection import train_test_split

        X = df.drop('diagnosis',axis=1)
        X = X.drop('Unnamed: 32',axis=1)
        y = df['diagnosis']
        # dividir datos
        train, test, train_labels, test_labels = train_test_split(X, y,
                                                  test_size = 0.33, random_state = 3)

In [10]: train.head()

Out[10]:           radius_mean  texture_mean  perimeter_mean  area_mean  \
         id
         917896          13.71         18.68           88.73      571.0
         8611792         19.10         26.29          129.10     1132.0
         864877          15.78         22.91          105.70      782.6
         904689          12.96         18.29           84.18      525.2
         89382602        12.76         13.37           82.29      504.1

                   smoothness_mean  compactness_mean  concavity_mean  \
         id
         917896            0.09916           0.10700         0.05385
         8611792           0.12150           0.17910         0.19370
```

```
           0.11550          0.17520         0.21330
           0.07351          0.07899         0.04057
           0.08794          0.07948         0.04052

          concave points_mean  symmetry_mean  fractal_dimension_mean  ...  \
id                                                                    ...
917896                0.03783         0.1714                 0.06843  ...
8611792               0.14690         0.1634                 0.07224  ...
864877                0.09479         0.2096                 0.07331  ...
904689                0.01883         0.1874                 0.05899  ...
89382602              0.02548         0.1601                 0.06140  ...

          radius_worst  texture_worst  perimeter_worst  area_worst  \
id
917896           15.11          25.63            99.43       701.9
8611792          20.33          32.72           141.30      1298.0
864877           20.19          30.50           130.30      1272.0
904689           14.13          24.61            96.31       621.9
89382602         14.19          16.40            92.04       618.8

          smoothness_worst  compactness_worst  concavity_worst  \
id
917896             0.14250             0.2566           0.1935
8611792            0.13920             0.2817           0.2432
864877             0.18550             0.4925           0.7356
904689             0.09329             0.2318           0.1604
89382602           0.11940             0.2208           0.1769

          concave points_worst  symmetry_worst  fractal_dimension_worst
id
917896                 0.12840          0.2849                  0.09031
8611792                0.18410          0.2311                  0.09203
864877                 0.20340          0.3274                  0.12520
904689                 0.06608          0.3207                  0.07247
89382602               0.08411          0.2564                  0.08253

[5 rows x 30 columns]
```

## 2 Evaluación de modelos

- Se obtienen las predicciones, informe de clasificación y matriz de confusión.
- Se crea lista para guardar evaluaciones

```
In [11]: from sklearn.metrics import confusion_matrix
         from sklearn.metrics import accuracy_score
```

```
In [12]: ev = []
```

## 2.1 BernoulliNB

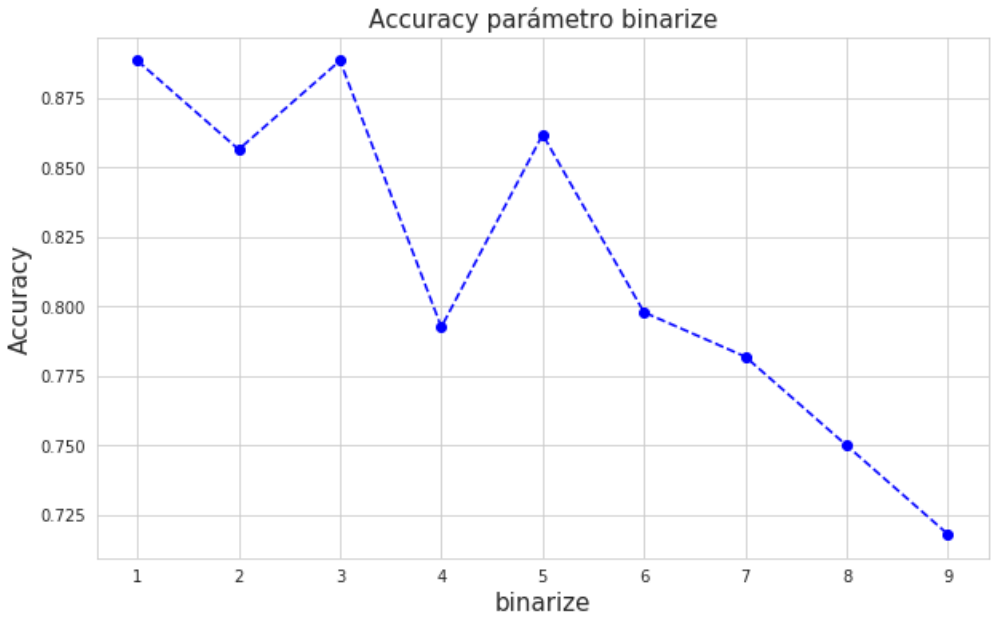
- Se ajusta el parámetro 'binarize'

```
In [13]: from sklearn.naive_bayes import BernoulliNB
```

```
In [14]: acc = []


         for b in range(1,10):

             bnb = BernoulliNB(binarize=0.1*b)
             bnb.fit(train,train_labels)
             pred = bnb.predict(test)
             acc.append(accuracy_score(test_labels, pred))
```

```
In [15]: plt.figure(figsize=(10,6))
         plt.plot(range(1,10),acc,color='blue', linestyle='--', marker='o')
         plt.title('Accuracy parámetro binarize', fontsize=15)
         plt.xlabel('binarize',fontsize=15)
         plt.ylabel('Accuracy',fontsize=15)
```

```
Out[15]: Text(0, 0.5, 'Accuracy')
```




- Con 'binarize' con valor de 0.3 se obtiene el mejor resultado

```
In [16]: bnb = BernoulliNB(binarize=0.3)
         bnb.fit(train, train_labels)
```

```
ev.append(bnb.score(test, test_labels))
bnb.score(test, test_labels)
```

```
Out[16]: 0.8882978723404256
```

```
In [17]: from sklearn.metrics import classification_report
```

```
In [18]: predictions = bnb.predict(test)
         print("Predicciones:\n")
         print(predictions)
         print("\nReporte de clasificación:\n")
         print(classification_report(predictions,test_labels))
```

```
Predicciones:

[0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0
 0 1 1 1 0 0 0 0 1 0 0 1 0 1 0 0 1 1 0 0 0 0 0 0 0 1 0 0 0 1 0 1 0 0 1 1 1
 0 1 0 1 1 1 1 0 1 0 0 1 0 0 0 0 1 1 1 0 0 1 0 1 0 0 0 0 0 0 0 1 1 0 0 0 0
 1 0 0 1 0 0 1 1 1 0 0 1 0 1 0 1 1 1 1 0 0 1 0 0 0 0 0 1 0 0 0 0 1 0 0 1 0 0
 1 0 1 1 0 0 0 1 1 0 0 0 0 0 1 0 0 1 0 0 0 0 0 0 0 0 1 1 0 1 0 0 0 0 1 1 0 0
 0 0 0]

Reporte de clasificación:

              precision    recall  f1-score   support

           0       0.95      0.88      0.92       129
           1       0.78      0.90      0.83        59

    accuracy                           0.89       188
   macro avg       0.86      0.89      0.88       188
weighted avg       0.90      0.89      0.89       188
```

```
In [19]: print("Confusion matrix")
         conf_mat=confusion_matrix(predictions,test_labels)
         print(conf_mat)
```

```
Confusion matrix
[[114  15]
 [  6  53]]
```

## 2.2 ComplementNB

```
In [20]: from sklearn.naive_bayes import ComplementNB
```

```
In [21]: cnb = ComplementNB()
         cnb.fit(train, train_labels)
         ev.append(cnb.score(test, test_labels))
         cnb.score(test, test_labels)
```

```
Out[21]: 0.898936170212766
```

```
In [22]: predictions = cnb.predict(test)
         print("Predicciones:\n")
         print(predictions)
         print("\nReporte de clasificación:\n")
         print(classification_report(predictions,test_labels))
```

```
Predicciones:

[0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 0 0 0 1 1 0 1 0 0 0 0 0 0
 0 1 1 1 0 0 1 0 0 0 0 1 0 1 0 0 1 1 0 0 1 0 0 0 0 1 0 0 0 1 0 1 1 0 0 1 1
 0 1 0 1 0 1 1 0 0 0 0 0 0 0 0 0 1 1 1 1 0 1 0 1 0 0 0 0 0 0 0 1 0 0 1 0 0
 1 0 0 1 1 0 1 1 1 0 0 1 0 0 0 1 0 0 0 0 0 1 0 0 0 0 1 0 1 0 0 1 0 0 1 0 0
 1 0 0 1 0 0 0 0 1 0 1 0 0 1 1 0 0 0 0 1 0 0 0 0 0 1 1 0 1 0 0 0 0 1 1 0 0
 1 0 0]

Reporte de clasificación:

              precision    recall  f1-score   support

           0       0.96      0.89      0.92       129
           1       0.79      0.92      0.85        59

    accuracy                           0.90       188
   macro avg       0.88      0.90      0.89       188
weighted avg       0.91      0.90      0.90       188
```

```
In [23]: print("Confusion matrix")
         conf_mat=confusion_matrix(predictions,test_labels)
         print(conf_mat)
```

```
Confusion matrix
[[115  14]
 [  5  54]]
```

## 2.3 GaussianNB

```
In [24]: from sklearn.naive_bayes import GaussianNB

         gnb = GaussianNB()

         gnb.fit(train, train_labels)
         ev.append(gnb.score(test, test_labels))
         gnb.score(test, test_labels)
```

```
Out[24]: 0.9574468085106383
```

```
In [25]: predictions = gnb.predict(test)
         print("Predicciones:\n")
         print(predictions)
         print("\nReporte de clasificación:\n")
         print(classification_report(predictions,test_labels))
```

```
Predicciones:

[0 0 0 0 1 0 0 0 0 0 0 1 0 1 0 0 0 0 1 0 0 0 0 1 0 0 0 1 1 0 1 0 0 0 0 0 0
 0 1 1 1 0 0 1 0 0 0 0 1 0 1 0 0 1 1 0 0 1 0 0 0 0 1 0 0 0 1 0 1 1 0 0 1 1
 0 1 0 1 1 1 1 0 0 0 0 1 0 0 0 0 1 1 1 0 0 1 0 1 0 0 0 0 0 0 0 1 1 0 1 0 0
 1 0 0 1 1 0 1 1 1 0 0 1 0 1 0 1 0 1 0 0 1 1 0 0 0 0 1 0 0 0 0 1 0 0 1 0 0
 1 0 0 1 0 0 0 1 1 0 1 0 0 0 1 0 0 1 0 1 0 0 0 0 0 1 1 0 1 0 0 0 0 1 1 0 0
 1 0 0]

Reporte de clasificación:

              precision    recall  f1-score   support

           0       0.97      0.96      0.97       122
           1       0.93      0.95      0.94        66

    accuracy                           0.96       188
   macro avg       0.95      0.96      0.95       188
weighted avg       0.96      0.96      0.96       188
```

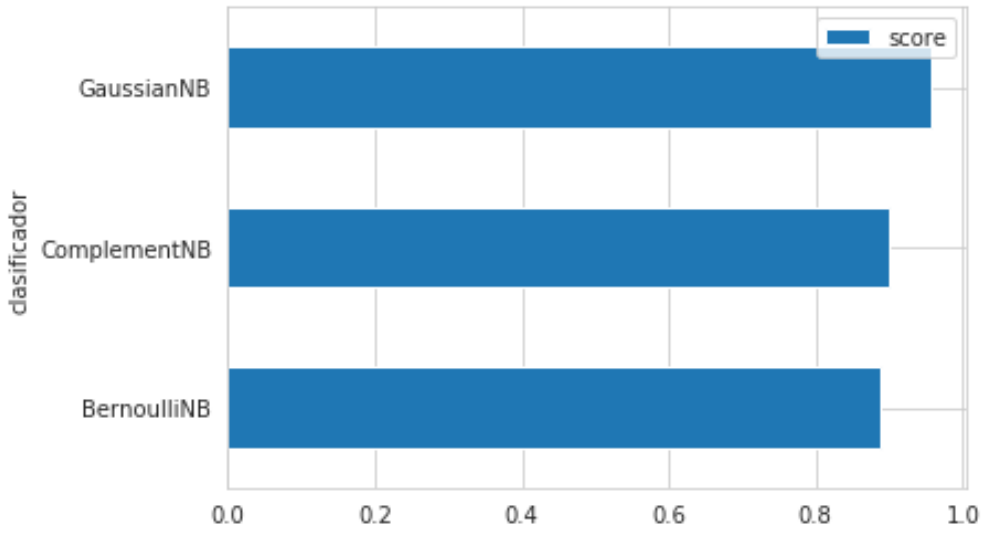
```
In [26]: print("Confusion matrix")
         conf_mat=confusion_matrix(predictions,test_labels)
         print(conf_mat)
```

```
Confusion matrix
[[117   5]
 [  3  63]]
```

```
In [27]: df = pd.DataFrame({'clasificador':['BernoulliNB','ComplementNB', 'GaussianNB'], 'score':
         df
```

```
Out[27]:    clasificador     score
         0   BernoulliNB  0.888298
         1  ComplementNB  0.898936
         2    GaussianNB  0.957447
```

```
In [28]: ax = df.plot.barh(x='clasificador', y='score')
```




- Usar el clasificador en otro dataset