

# Métricas para clasificación

The background of the slide is a photograph of a clear night sky. The Milky Way galaxy is visible as a bright, hazy band of light stretching across the upper half of the image. Numerous individual stars are scattered throughout the dark blue sky. In the lower portion of the image, the dark, silhouetted branches of evergreen trees are visible, framing the bottom of the scene.

# Métricas para clasificación binaria

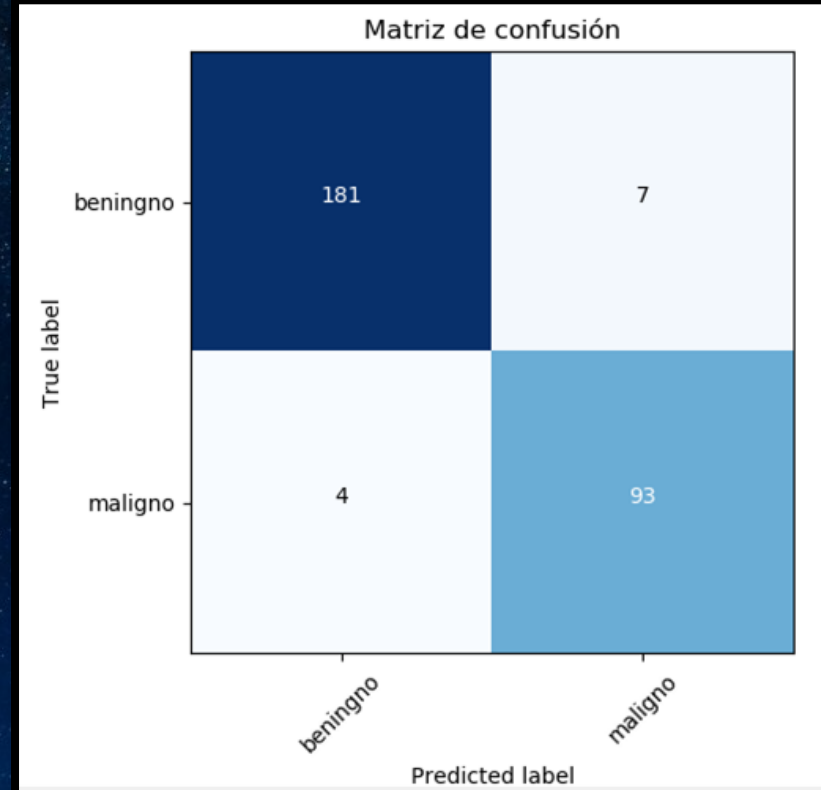
- El error (entropía cruzada o MSE) nos permite saber cómo se ajustan las probabilidades del modelo pero no nos explica cuan bien clasifica el modelo.
- El Accuracy solo sirve como promedio de clasificación general del modelo.
- Veremos a continuación una serie de métricas que nos permitirán observar con más detalle este comportamiento.



# Matriz de confusión (clasificación binaria)

- Una matriz de confusión permite visualizar el funcionamiento del modelo de clasificación.
- Las filas representan la etiqueta real de cada clase, mientras que las columnas representan la predicción del modelo entrenado.
- La diagonal principal representa el correcto funcionamiento del modelo. Fuera de la diagonal: errores.

```
AAutils.plot_confusion_matrix(y_true, y_pred)
```



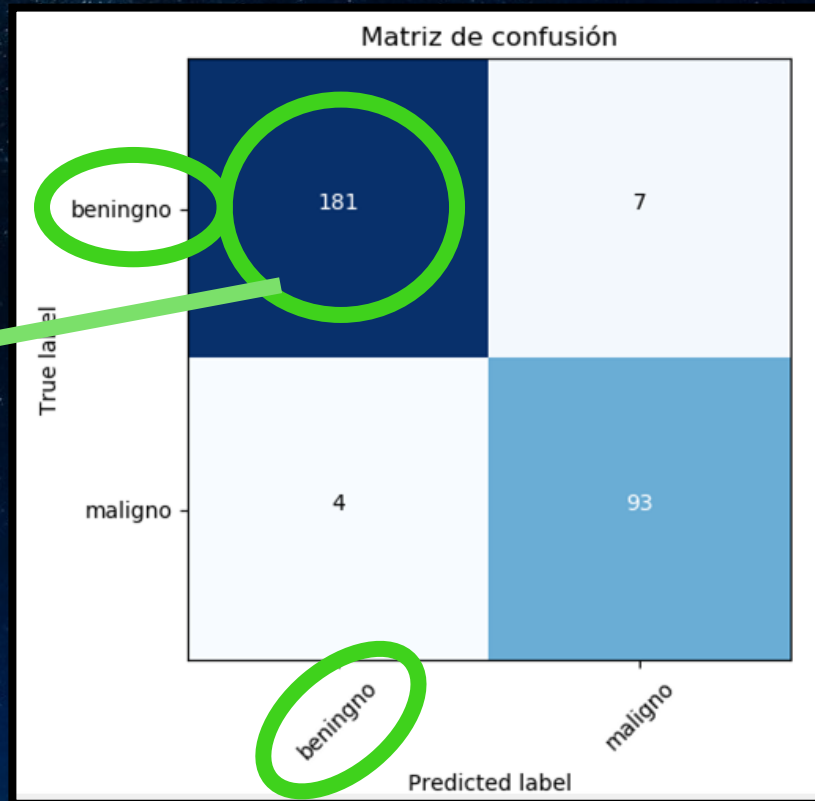
# Matriz de confusión (clasificación multiclase)

Misma idea para multiclase.  
Por ahora seguiremos con  
Clasificación Binaria.

True label	airplane	automobile	bird	cat	deer	dog	frog	horse	ship	truck
	29	6	5	0	3	1	2	4	15	6
	4	47	0	0	1	0	4	2	7	8
	12	0	11	7	9	11	13	10	1	4
	2	4	6	9	10	21	11	10	3	4
	11	1	5	6	20	7	10	15	0	8
	2	1	5	12	5	25	8	12	0	5
	0	6	7	3	2	4	47	6	2	2
	3	2	5	4	7	6	2	48	3	5
	13	17	4	1	2	0	0	6	37	5
Predicted label	airplane	automobile	bird	cat	deer	dog	frog	horse	ship	truck
	4	14	0	6	0	3	1	5	3	55

# Matriz de confusión (clasificación binaria)

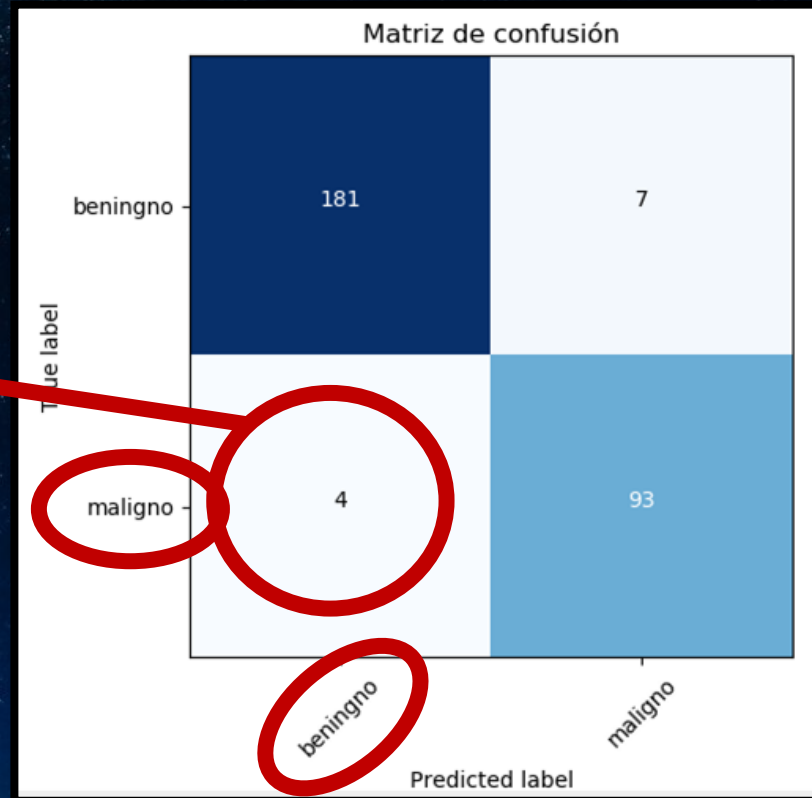
181 ejemplos eran Benignos y el modelo los clasificó como Benignos.



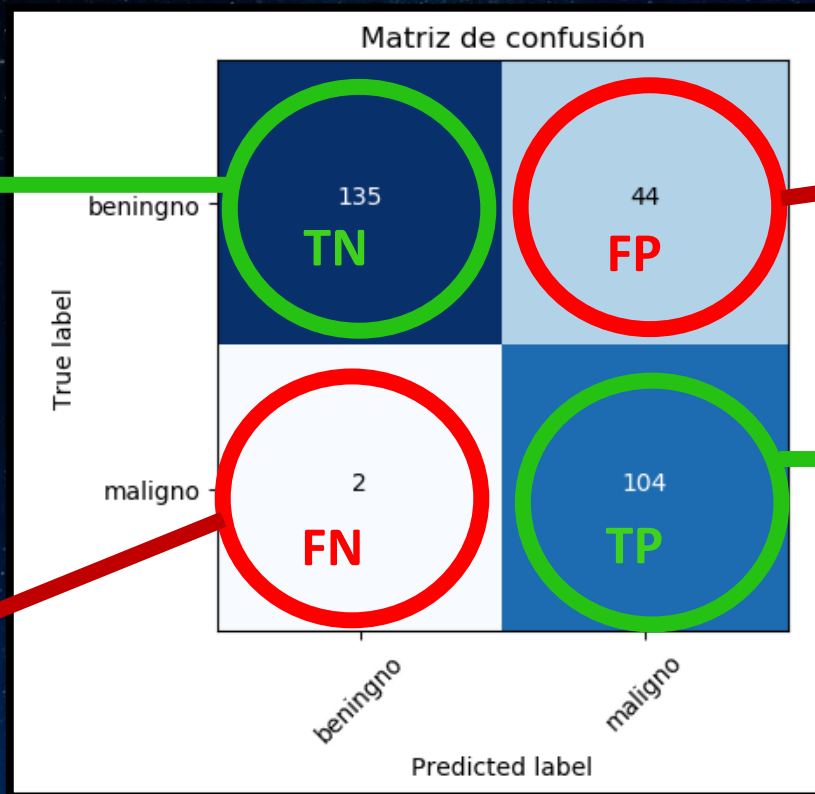


# Matriz de confusión (clasificación binaria)

4 ejemplos eran  
Malignos y el modelo  
los clasificó como  
Benignos (Error).



# Métricas



True Negatives

Falses Positives

Falses Negatives

True Positives

# Métricas - Accuracy

$$\text{Accuracy} = \frac{\text{num prediccc. correctas}}{\text{total predicciones}} = \frac{TP + TN}{(TP + TN + FP + FN)}$$

$$\text{Acc} = \frac{1}{n} \sum_i^n (y^{\wedge}_i - y_i)$$

$$\text{Acc} = (104 + 135) / (104 + 135 + 44 + 2) = \mathbf{0,84}$$

Matriz de confusión

True label	Predicted label	
	benigno	maligno
benigno	135	44
maligno	2	104

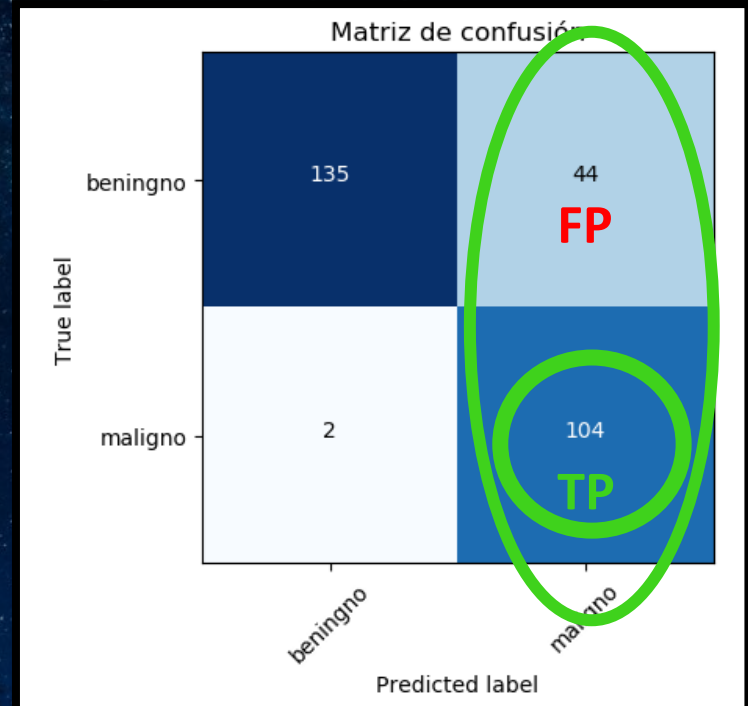


# Métricas - Precision

$$\text{Precision} = \frac{TP}{(TP+FP)}$$

$$\text{Prec} = 104 / (104+44) = 0,70$$

La **precisión** nos dice cuántos ítems reconocidos son realmente relevantes

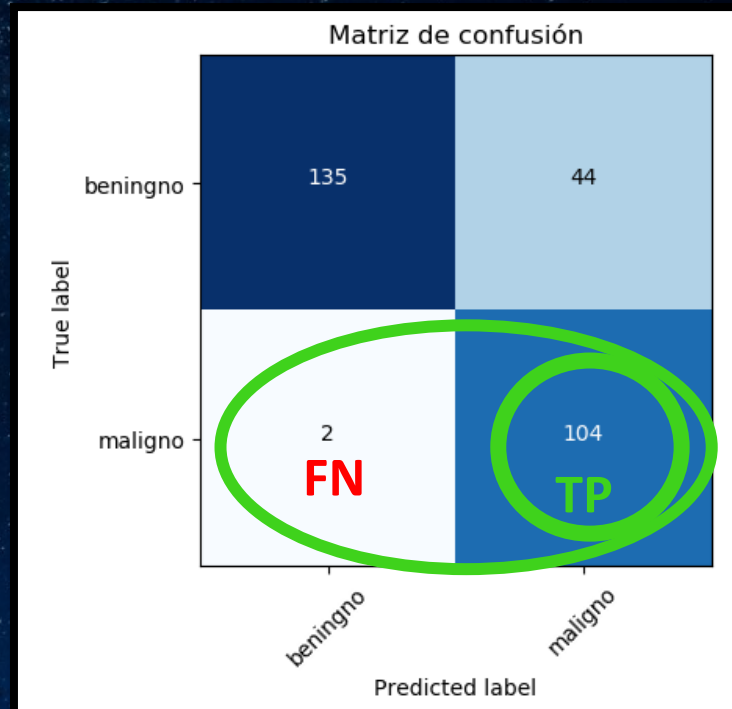


# Métricas - Recall

$$\text{Recall} = \frac{TP}{(TP+FN)}$$

$$\text{Rec} = 104 / (104+2) = 0,98$$

El **recall** nos dice cuántos ítems relevantes fueron realmente seleccionados



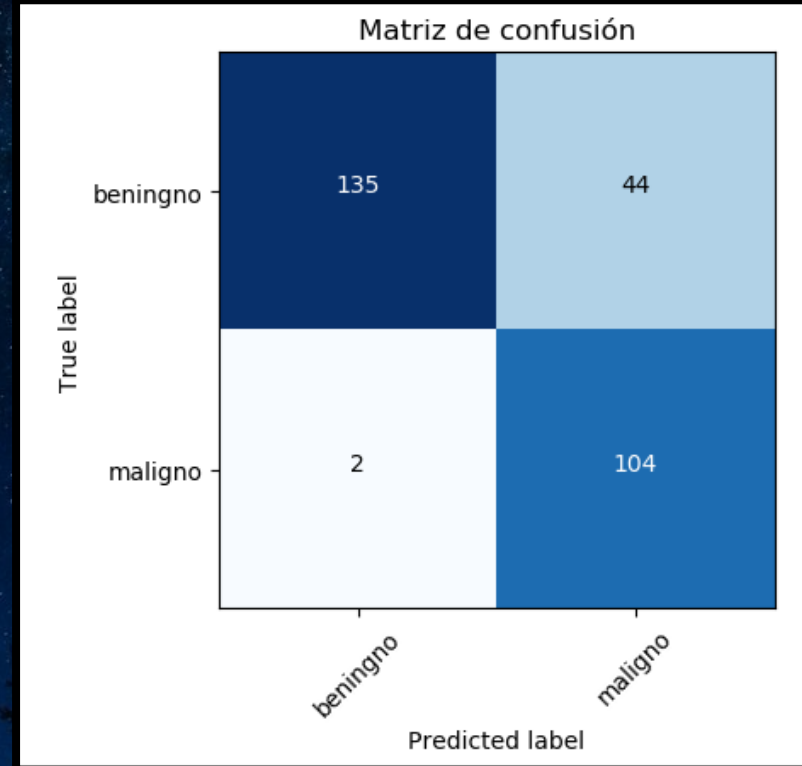
# Métricas - Resumen

**Acc = 0,84**

**Prec = 0,70**

**Rec = 0,98**

- El Acc. nos muestra cómo se comporta el modelo teniendo en cuenta tanto los TP como los TN.
- El Prec. Y Rec. solo tienen en cuenta los TP.
- En clasificación binaria generalmente nos va a interesar más el Prec. Y Rec.





# Métricas – F-measure

Para encontrar un balance entre estas métricas, suele utilizarse la media armónica, también conocida como f-measure (o F1-score):

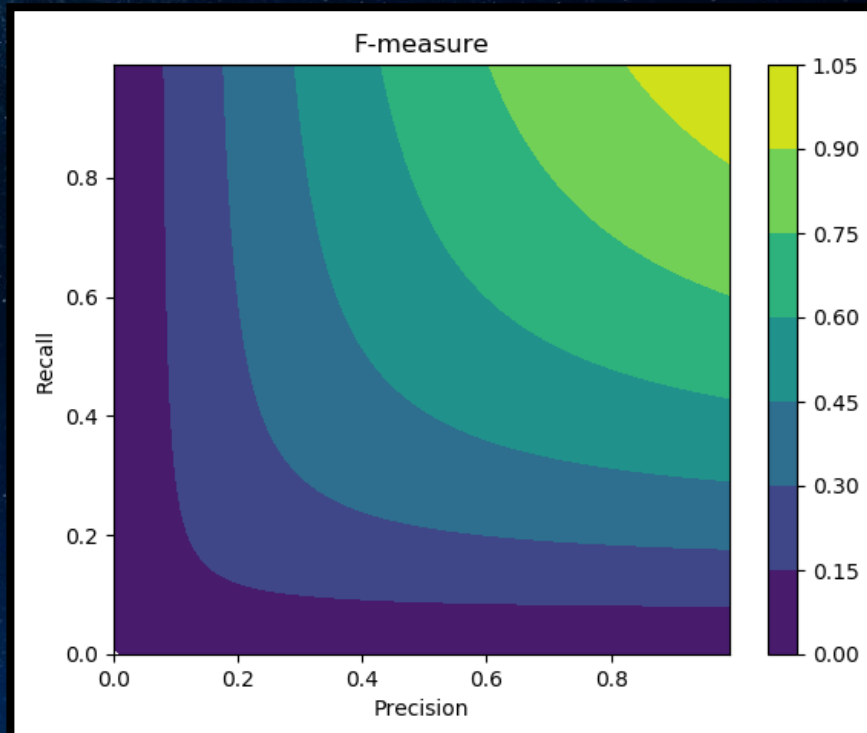
$$f - measure = 2 \times \frac{precision \times recall}{precision + recall}$$

Acc = 0,84

Prec = 0,70

Rec = 0,98

**F-measure = 0,82**



# Métricas – Multiclase

Para un problema multiclase es posible calcular las métricas de forma individual.

```
from sklearn.metrics import f1_score,  
    recall_score,  
    precision_score,  
    accuracy_score,  
    classification_report  
  
print(classification_report(y_test, y_pred))
```

	precision	recall	f1-score	support
0.0	0.98	0.99	0.99	196
1.0	0.99	0.97	0.98	198
2.0	0.99	0.96	0.98	197
3.0	1.00	1.00	1.00	207
4.0	0.88	0.88	0.88	206
5.0	0.88	0.90	0.89	196
accuracy			0.95	1200
macro avg	0.95	0.95	0.95	1200
weighted avg	0.95	0.95	0.95	1200

# Ejemplos métricas – Dataset desbalanceado

Acc = 0,98

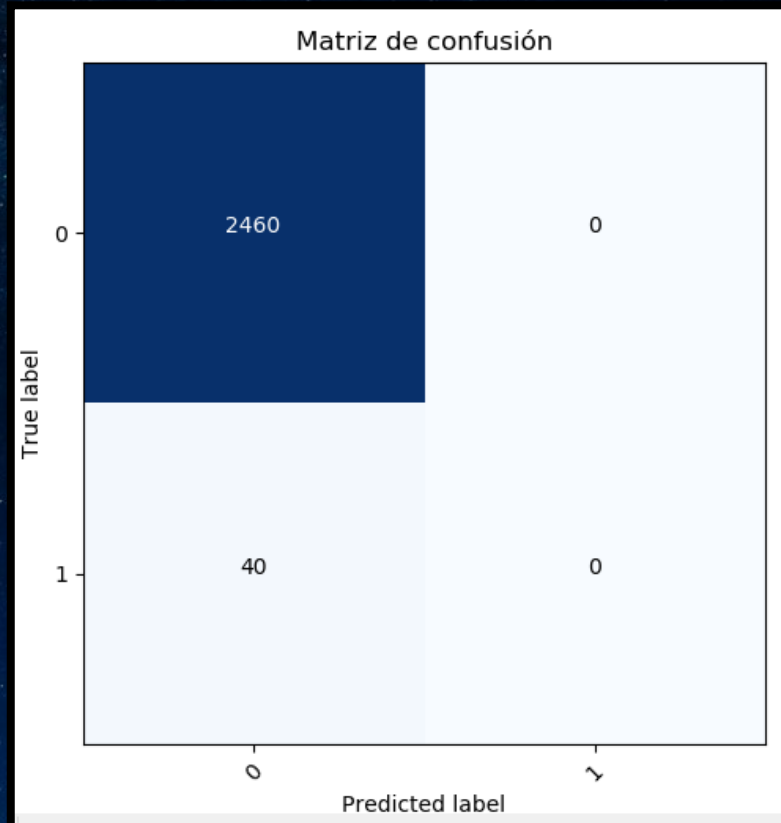
Prec = 0

Rec = 0

F-measure = 0

En este caso el dataset está muy desbalanceado (una clase es mayoritaria), lo que ocasiona que el modelo “aprenda” a decir siempre lo mismo.

Recordar que el modelo minimiza el error medio de todos los ejemplos.



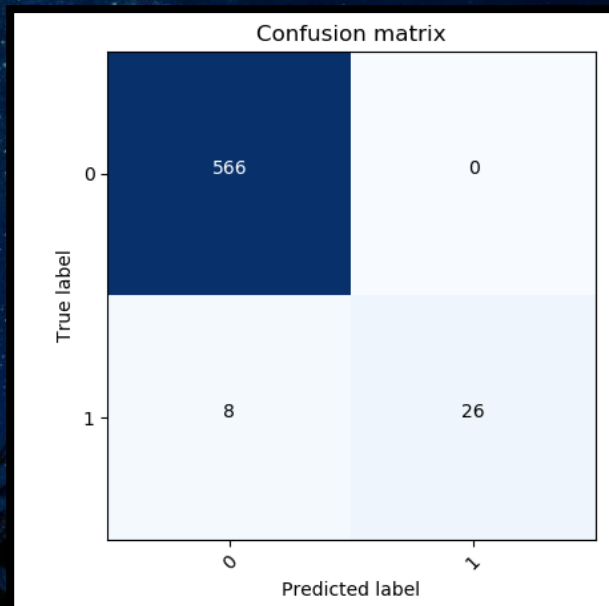


# Dataset desbalanceado

Tenemos dos modos de solucionar el problema del desbalance de clases:

- Uno sería implementar nuestra propia función de error para que tenga en cuenta otra métrica.
- Otro es con el parámetro `class_weight` al momento de entrenar el modelo. Este parámetro es un diccionario que indica el peso que se le dará a cada clase dentro de la función de error durante el entrenamiento.

```
Test Accuracy: 0.99  
Precision: 1.00  
Recall: 0.76  
f-measure: 0.87
```



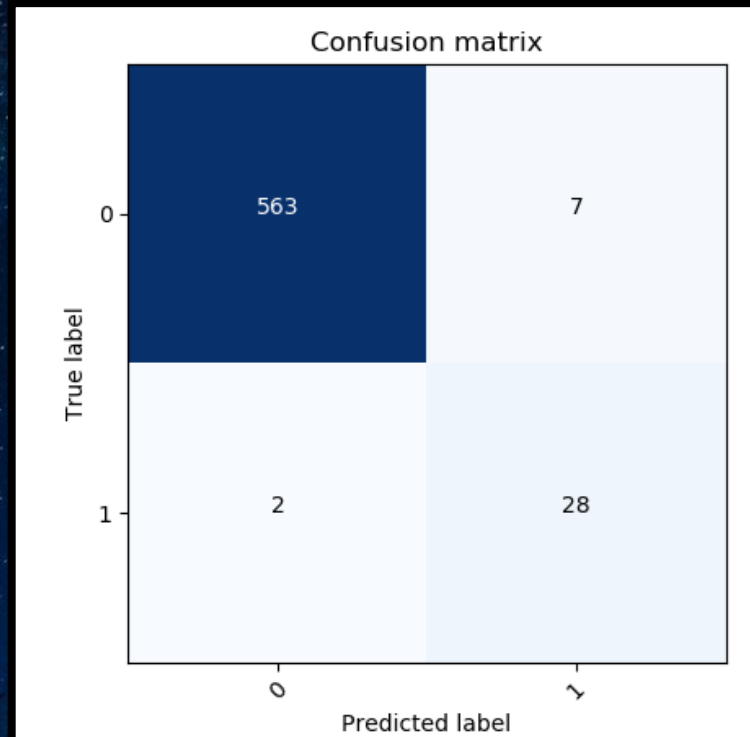
# Dataset desbalanceado

```
Test Accuracy: 0.98  
Precision: 0.80  
Recall: 0.93  
f-measure: 0.86
```

```
model.fit(X_train,  
          Y_train,  
          class_weight= {0:0.3, 1:0.7} )
```

A medida que inclinamos el peso hacia una u otra clase, aumentará el Recall y disminuirá el Precision, o viceversa.

Nota: En SkLearn, para Redes Neuronales no está implementado el parámetro.



# Dataset desbalanceado

Test    Accuracy: 0.99  
         Precision: 1.00  
         Recall: 0.76  
         f-measure: 0.87

`class_weight= {0:0.5, 1:0.5}`

Test    Accuracy: 0.98  
         Precision: 0.80  
         Recall: 0.93  
         f-measure: 0.86

`class_weight= {0:0.3, 1:0.7}`

Test    Accuracy: 0.95  
         Precision: 0.60  
         Recall: 0.95  
         f-measure: 0.73

`class_weight= {0:0.05, 1:0.95}`

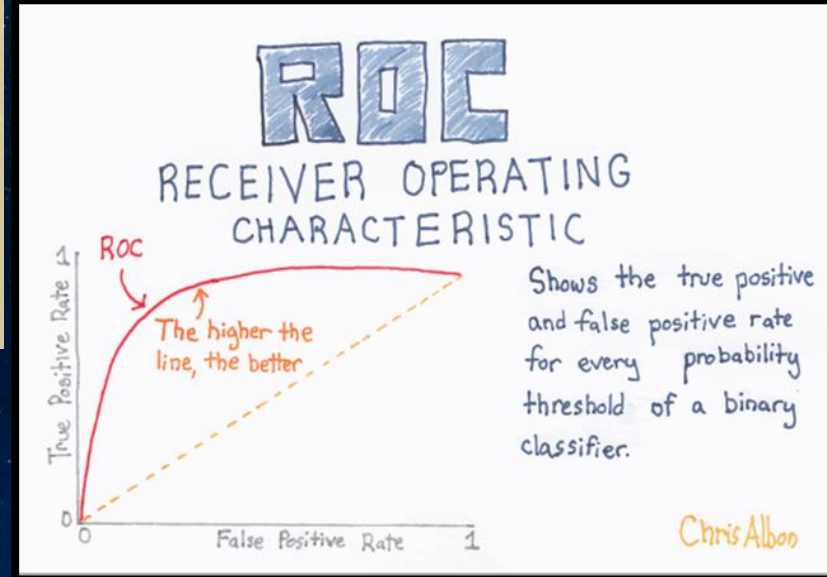
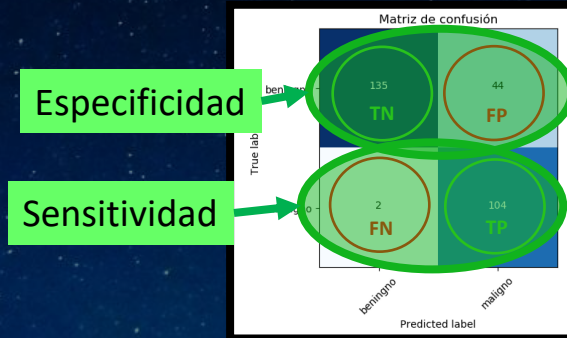


# Curvas ROC

Las curvas ROC (Receiver Operating Characteristic) nos muestran cómo se comporta un modelo binario al cambiar el umbral de detección.

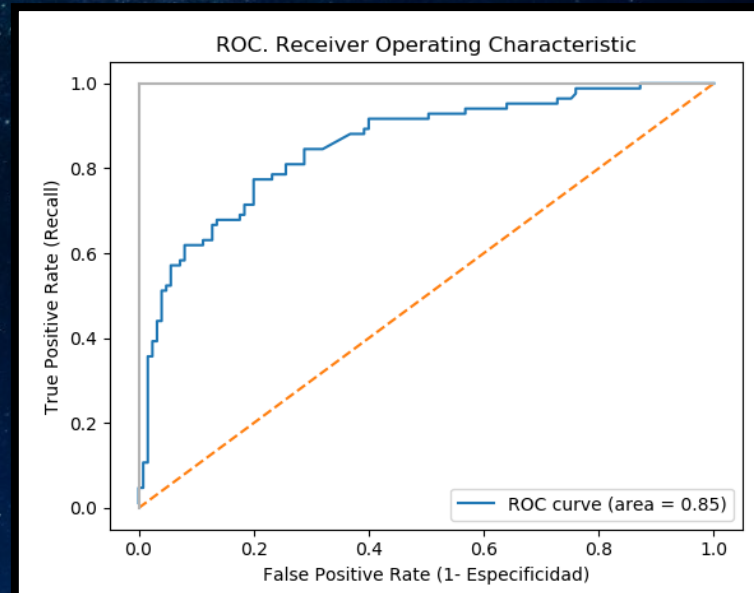
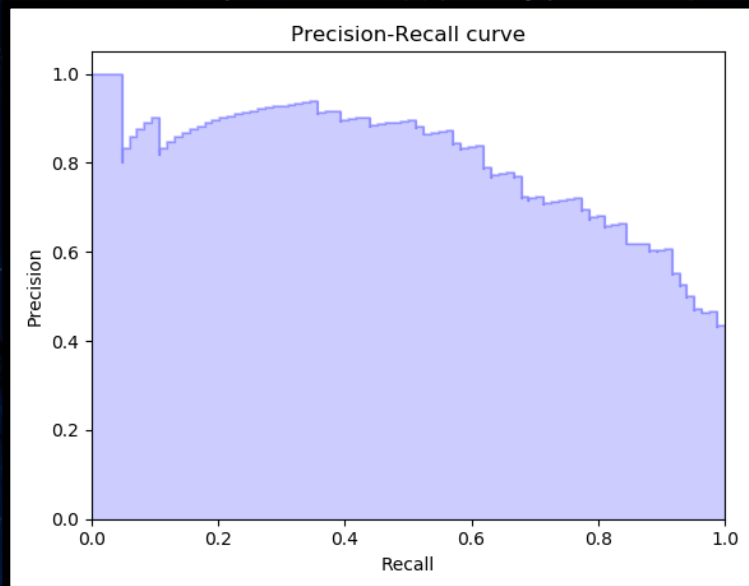
**True Positive Rate** = Sensitividad = Recall =  $TP / (TP + FN)$

**False Positive Rate** = 1 - Especificidad =  $1 - TN / (FP + TN)$



# Curvas Precision-Recall

Las curvas **Precision-Recall** se computan de igual modo que las curvas ROC pero grafican el funcionamiento del modelo para estas métricas.



```
AAutils.plot_ROC_curve(model, x, y)
```