Título: Trabajo Aprendizaje Automático I

Máster Universitario en Informática Industrial y Robótica

Alumnos:

Domingo Alberto Capelo Luces

Iraisy Carolina Figueroa Silva

Fecha: 19 abril del 2021

INDICE

Descripción de los datos empleado

El punto de partida consta de un banco de 5983 imágenes y un archivo Data.csv.

En lo referente a las imágenes se cumple que:

- Nombre: image**#N**.jpg-→ **#N**: número de imagen.
- Imagen a color RGB.
- Dimensiones distintas.

Ejemplo:

```
Imagen 0: image10.jpg
Imagen 1: image10000.jpg
Imagen 2: image10002.jpg
Imagen 3: image10003.jpg
Imagen 4: image10004.jpg
```

En lo referente al archivo Data.csv, se clasifican las imágenes en 4 grupos:

'Food', 'Attire', 'Decorationandsignage', 'misc'.

Ejemplo:

Image	**Class**
image7042.jpg	Food
image3327.jpg	Misc
image10335.jpg	Attire
image8019.jpg	Food
image2128.jpg	Attire
image1106.jpg	Misc
image6750.jpg	Food

Partiendo de estos datos y con el uso del lenguaje de programación Python, se realizará el entrenamiento y posterior comparación de modelos de aprendizaje automático aplicados a la clasificación de imágenes.

El primer paso, antes de entrenar cualquier modelo, es preparar las variables que vamos a introducir como entrada.

Tratamiento entradas y salidas

Objetivo:

x: entrada del modelo, matriz de bits de una imagen.

t: salida del modelo, clasificación entrada.

x → MODELO →t

PASO 1: Redimensión imagenes

Las imágenes no tienen la misma dimensión, por lo que se redimensionan para que todas tengan la misma: 64x64 Pixeles.

Imágenes originales:

```
image10.jpg--> pixels:(60, 80, 3)
image10000.jpg--> pixels:(53, 80, 3)
image10002.jpg--> pixels:(107, 80, 3)
image10003.jpg--> pixels:(65, 80, 3)
image10004.jpg--> pixels:(58, 80, 3)
```

Calidad 100%
image10.jpg

Calidad 100%
image10000.jpg

Imágenes redimensionadas:

```
image10.jpg--> pixels redim: (64, 64, 3)
image10000.jpg--> pixels redim: (64, 64, 3)
image10002.jpg--> pixels redim: (64, 64, 3)
image10003.jpg--> pixels redim: (64, 64, 3)
image10004.jpg--> pixels redim: (64, 64, 3)
```

Redim 64x64
image10.jpg

Redim 64x64
image10000.jpg

PASO 2 : Asociacion clase imagen

La clasificación de la imagen que se representa en el archivo Data.csv se asocia el nombre de una imagen con una clase. En nuestro caso nos interesa asociar la matriz de pixeles de cada imagen con su clase.

imagen.jpg →clase

Image	Class
image4707.jpg	Attire
image9827.jpg	misc
image8322.jpg	Food
image8426.jpg	Food
image2346.jpg	Attire
...	...
image4667.jpg	Decorationandsignage
image2029.jpg	Food
image7113.jpg	misc
image10833.jpg	Food
image8858.jpg	Attire

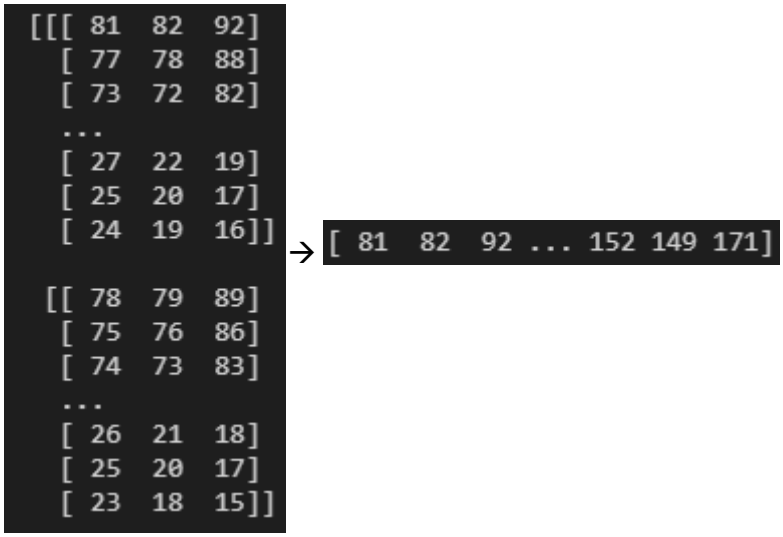
Matriz píxeles →clase

Image	Class
(64, 64, 3)	['Attire']
(64, 64, 3)	['misc']
(64, 64, 3)	['Food']
(64, 64, 3)	['Food']
(64, 64, 3)	['Attire']
(64, 64, 3)	['Attire']
(64, 64, 3)	['Decorationandsignage']
(64, 64, 3)	['Attire']
(64, 64, 3)	['Food']
(64, 64, 3)	['misc']

PASO 3 : Aplanamiento matriz

Las matrices de pixeles no se pueden introducir como entradas a los modelos directamente, hay que realizar un proceso de aplanamiento de la matriz de forma que se representen todos los pixeles de la imagen en un único vector.

(64,64,3) (1, 12288)



De esta forma para las 5983 imágenes tendremos todos los vectores de entrada de nuestro modelo.

En el siguiente ejemplo se muestran las primeras 11 entradas:

```
X0: [ 81  82  92 ... 152 149 171]
X1: [0 0 0 ... 2 3 0]
X2: [26 40 29 ... 35 47 51]
X3: [0 0 3 ... 0 0 0]
X4: [163 178 174 ...  42  40  30]
X5: [  7   0   7 ...  27  45 198]
X6: [167 155 155 ...  33  22 119]
X7: [ 41  37 196 ...  41  15 183]
X8: [ 45 108 206 ... 106 131 103]
X9: [240 226 255 ... 125 128 132]
X10: [255 255 255 ... 255 251 255]
```

PASO 4 : Normalización entradas

Para miminizar el tiempo de entrenamiento y debido a que los datos tienen una variación de 0 a 255 ya que son de 8 bits, se opta por normalizar las entradas de forma que varien entre 0 y 1 y de esta forma lograr que el resultado del entrenamiento sea mas preciso.

Sin normalizar

```
X0: [ 81  82  92 ... 152 149 171]
X1: [0 0 0 ... 2 3 0]
X2: [26 40 29 ... 35 47 51]
X3: [0 0 3 ... 0 0 0]
X4: [163 178 174 ...  42  40  30]
X5: [  7   0   7 ...  27  45 198]
X6: [167 155 155 ...  33  22 119]
X7: [ 41  37 196 ...  41  15 183]
X8: [ 45 108 206 ... 106 131 103]
X9: [240 226 255 ... 125 128 132]
X10: [255 255 255 ... 255 251 255]
```

Normalizado

```
X0: [0.31764706 0.32156863 0.36078431 ... 0.59607843 0.58431373 0.67058824]
X1: [0.         0.         0.         ... 0.00784314 0.01176471 0.        ]
X2: [0.10196078 0.15686275 0.11372549 ... 0.1372549  0.18431373 0.2       ]
X3: [0.         0.         0.01176471 ... 0.         0.         0.        ]
X4: [0.63921569 0.69803922 0.68235294 ... 0.16470588 0.15686275 0.11764706]
X5: [0.02745098 0.         0.02745098 ... 0.10588235 0.17647059 0.77647059]
X6: [0.65490196 0.60784314 0.60784314 ... 0.12941176 0.08627451 0.46666667]
X7: [0.16078431 0.14509804 0.76862745 ... 0.16078431 0.05882353 0.71764706]
X8: [0.17647059 0.42352941 0.80784314 ... 0.41568627 0.51372549 0.40392157]
X9: [0.94117647 0.88627451 1.         ... 0.49019608 0.50196078 0.51764706]
X10: [1.         1.         1.         ... 1.         0.98431373 1.        ]
```

PASO 5: Normalización salidas

Para facilitar el entrenamiento de los modelos, las clases que tienen un formato del tipo “string” se asocian con un número.

'Food': 0, 'Attire': 1, 'Decorationandsignage':2, 'misc':3

Ejemplo:

```
Food: 0
misc: 3
Attire: 1
Food: 0
Attire: 1
misc: 3
Food: 0
Food: 0
Food: 0
Food: 0
```

```
t0: 0
t1: 3
t2: 1
t3: 0
t4: 1
t5: 3
t6: 0
t7: 0
t8: 0
t9: 0
t10: 1
```

Finalmente ya podemos realizar los entrenamientos de los modelos con el uso de nuestras entradas salidas previamente tratadas:

X normalizado

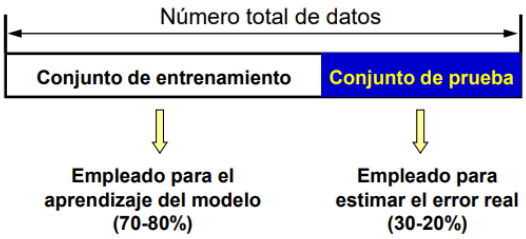
t numérica

En el siguiente estudio se analizaran los resultados de distintos modelos de aprendizaje automático supervisado aplicados a la clasificación de imágenes.

Cabe destacar que el aprendizaje es supervisado porque además de las entradas del modelo se indican las salidas del mismo.

Estrategia de evaluación de modelos

Una opción sería dividir los datos en entrenamiento y prueba, para poder evaluar los aciertos/fallos sobre los datos de prueba.



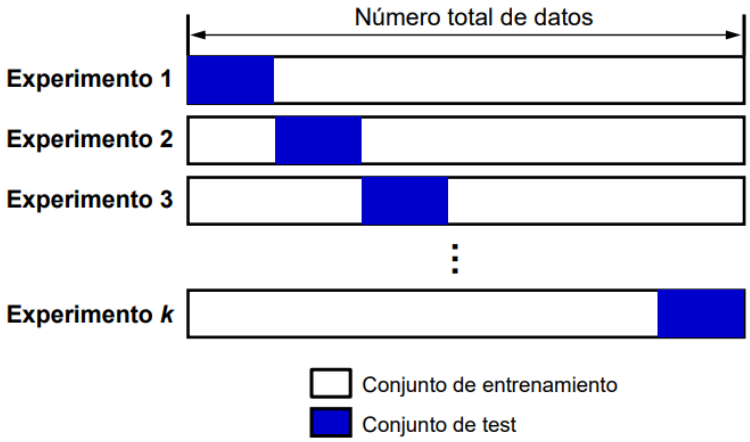
Pero, ya que:

- No disponemos de una parte importante como conjunto de pruebas para nuestro modelo.
- Se corre el riesgo de que al dividir los datos para Train/Test estén desbalanceados por lo que el entrenamiento del modelo y su posterior evaluación sobre los datos de Test no sea fiable.

Se opta por realizar:

K-fold cross-validation

Todas las muestras del conjunto de datos se usan alguna vez para entrenar o como parte del conjunto de prueba, esto soluciona la problemática anterior.



Para ello se utiliza la función cross_validate, que nos permite obtener los parámetros significativos de los modelos estudiados.

Los parámetros evaluados son los siguientes:

Exactitud (accuaracy): tasa de acierto global del sistema.

Roc Auc (roc_auc_ovo): El AUC calcula el área bajo la curva ROC y, por lo tanto, su valor varía entre 0 (peor caso) y 1 (mejor caso). El AUC indica cómo de bien se separan las probabilidades de las clases positivas de las negativas.

F1 macro: es la media armónica entre la precisión y la sensibilidad, tiene la ventaja que su métrica es más relevante si las clases son desbalanceadas.

Presición (precision_macro): de todos los ejemplos que se clasificaron como positivos, ¿cuántos son realmente positivos?.

Recall_macro: de todos los ejemplos positivos, ¿cuántos se pronostican como positivos?.

Modelos de aprendizaje supervisado aplicados a clasificación de imágenes

Modelos Lineales

Regresión lineal

- Se producen problemas especialmente cuando las clases están desbalanceadas.
- Problemas cuando hay clases no homogéneas (datos atípicos).
- Por lo general, no es un método muy adecuado para la clasificación.
- Método de mínimos cuadrados.

Regresión logística

- En general es el método más robusto ya que no es sensible a casos atípicos.
- Para solucionar el problema del método de mínimos cuadrados, la regresión logística propone el uso de una función logística o sigmoide.

¿Qué modelo de regresión utilizar?

Analizamos nuestros datos:

Como vemos en la siguiente imagen, partimos de datos desbalanceados, ya que el número de datos en las clases difieren entre ellos.

```
Clase ' Food ' hay 2278 imagenes
Clase ' Attire ' hay 1691 imagenes
Clase ' Decorationandsignage ' hay 743 imagenes
Clase ' misc ' hay 1271 imagenes
```

El caso más crítico es el de **'Food'** en el que tenemos **2278** imágenes frente a las **743** de **'Decorationanddsignage'**.

El modelo de regresión escogido evaluando las condiciones de nuestro problema es la regresión logística.

Regresión Logística:

Con el uso de la función LogisticRegression disponible en la librería sklearn, se realizó el entrenamiento del modelo de regresión logística.

A tener en cuenta:

Logistic Regression (aka logit, MaxEnt) classifier

"In the multiclass case, the training algorithm uses the one-vs-rest (OvR) scheme if the 'multi_class' option is set to 'ovr', and uses the cross-entropy loss if the 'multi_class' option is set to 'multinomial'. (Currently the 'multinomial' option is supported only by the 'lbfgs', 'sag', 'saga' and 'newton-cg' solvers.)"

solver*{'newton-cg', 'lbfgs', 'liblinear', 'sag', 'saga'}, default='lbfgs'*

"For multiclass problems, only 'newton-cg', 'sag', 'saga' and 'lbfgs' handle multinomial loss; 'liblinear' is limited to one-versus-rest schemes"

multi_class*{'auto', 'ovr', 'multinomial'}, default='auto'*

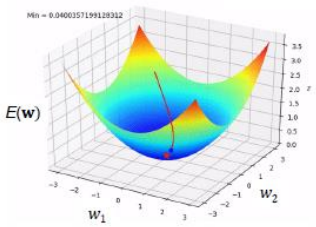
"If the option chosen is 'ovr', then a binary problem is fit for each label. For 'multinomial' the loss minimised is the multinomial loss fit across the entire probability distribution, *even when the data is binary*. 'multinomial' is unavailable when solver='liblinear'. 'auto' selects 'ovr' if the data is binary, or if solver='liblinear', and otherwise selects 'multinomial'"

max_iterint, default=100
"Maximum number of iterations taken for the solvers to converge."

Teniendo en cuenta lo anterior y a que nuestros datos no son binarios en la opción multi_class escogeremos 'multinomial' y como solver 'lbfgs' que es compatible con este tipo de clasificación.

Sintaxis:

El objetivo de las iteraciones es lograr que nuestro modelo converja: mínimo error /mejores pesos de entrenamiento de nuestro modelo.

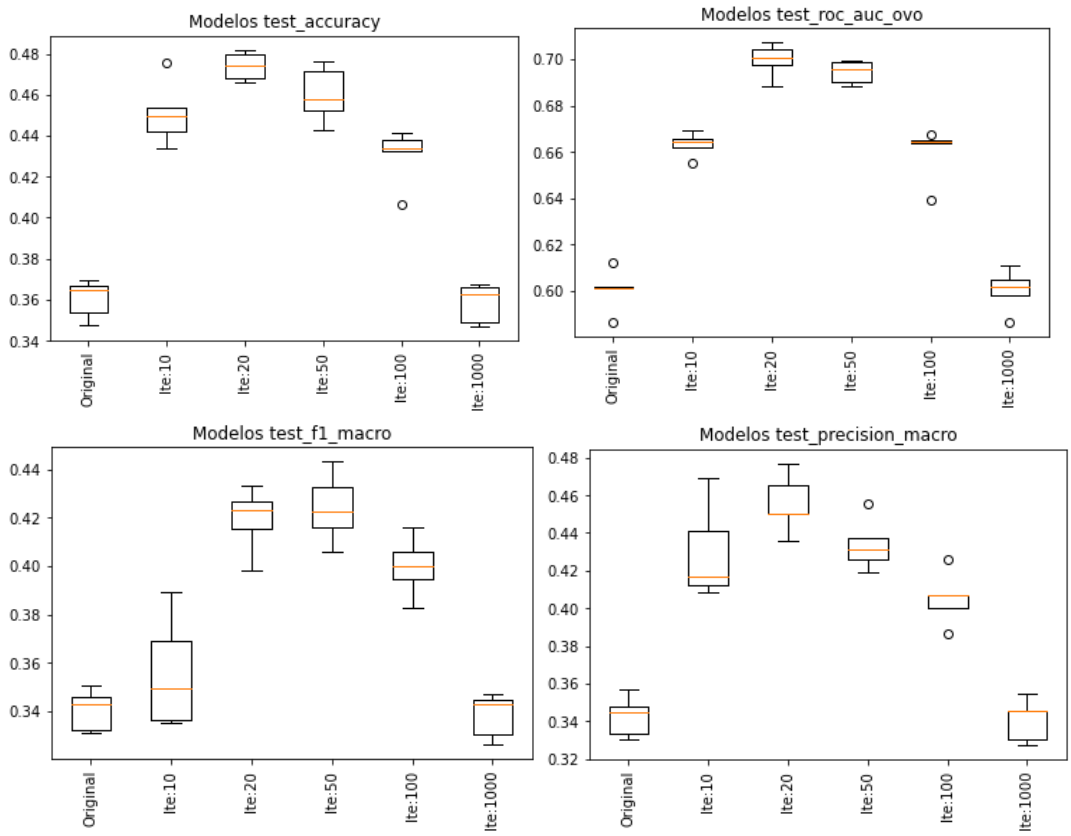


En nuestro caso escogimos 10000 iteraciones para llegar a la convergencia.

```
LogisticRegression(penalty='none', solver = 'lbfgs', max_iter=10000
,multi_class='multinomial')
```

Estudio mediante Hiperparametros LR:

Haremos un estudio de cuál es la diferencia en variar el número iteraciones en nuestro modelo.



Como observamos para valores de 10, 20, 50 iteraciones marca valores mejores de los parámetros que para valores mayores de iteración, esto es porque para esos valores el modelo se quedó en mínimos locales y es bueno para clasificar la clase mayoritaria por ejemplo 'food'.
Si dejamos que el modelo converja se evidencia el déficit del uso de este tipo de estrategias en la clasificación de imágenes.

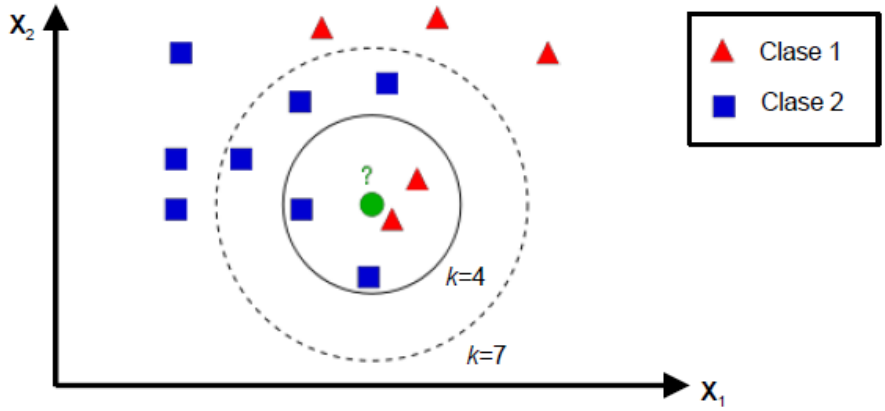
Modelos no lineales

k Nearest Neighbors KNN

Clasifica los datos tomando como referencia la clase mayoritaria de entre los k vecinos más cercanos de entre todos los datos de entrenamiento.

En nuestro caso, en base a las siguientes imágenes, las clasificara en base a la clase que tengan las imágenes más cercanas a ella.

```
Clase ' Food ' hay 2278 imagenes
Clase ' Attire ' hay 1691 imagenes
Clase ' Decorationandsignage ' hay 743 imagenes
Clase ' misc ' hay 1271 imagenes
```



Los métodos de este tipo emplean una métrica (medida disimilitud o distancia) entre los distintos datos.

Nuestro conjunto de datos de entrenamiento consta de 5983 imágenes por lo que según la teoría el valor de k tiene que ser menor a la raíz cuadrada de nuestro número de datos

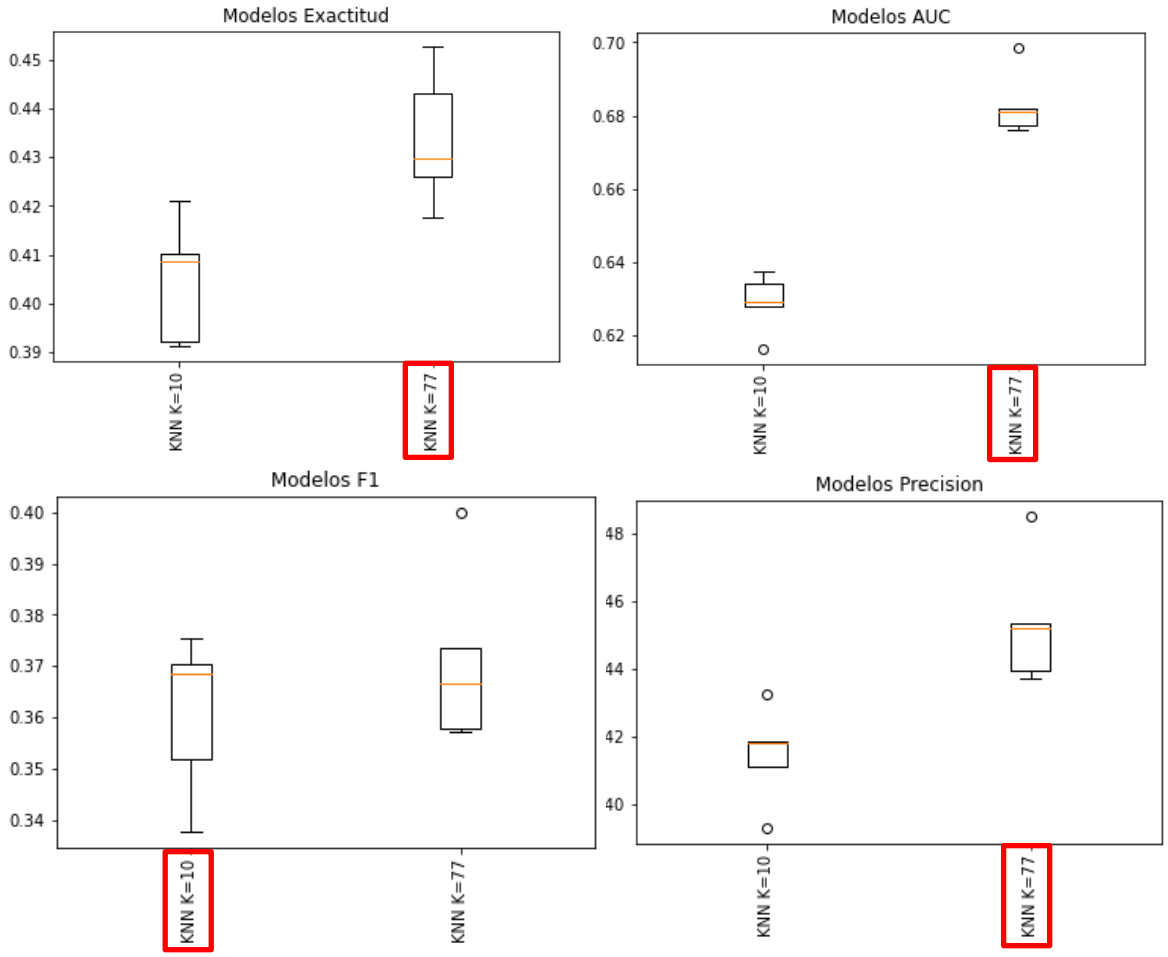
Sqrt(5983)=77,34

Como estrategia de evaluación del modelo al igual que en el modelo de regresión logística se opta por realizar validación cruzada evaluando los parámetros comentados anteriormente.

Estudio hiperparametros KNN:

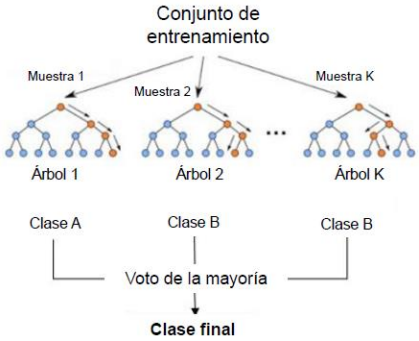
Haremos un estudio de cuál es la diferencia variar la k entre 10 y 77.

Como observamos en los boxplots, la mejor opción es con K=77 debido a que tiene mejores valores en la evaluación de la validación cruzada.



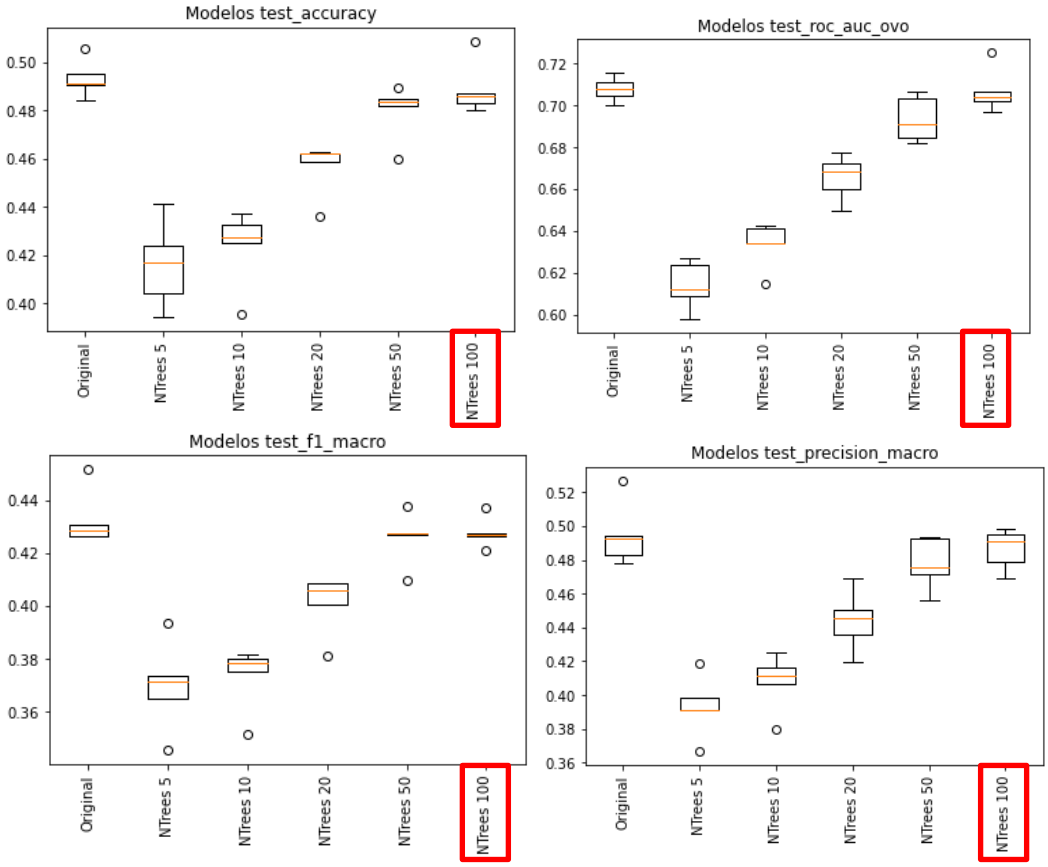
Random Forest

Formado por un conjunto de árboles de decisión individuales, cada uno entrenado con una muestra ligeramente distinta de los datos de entrenamiento emplea un modelo de clasificación en forma de árbol.



Estudio hiperparametros RF:

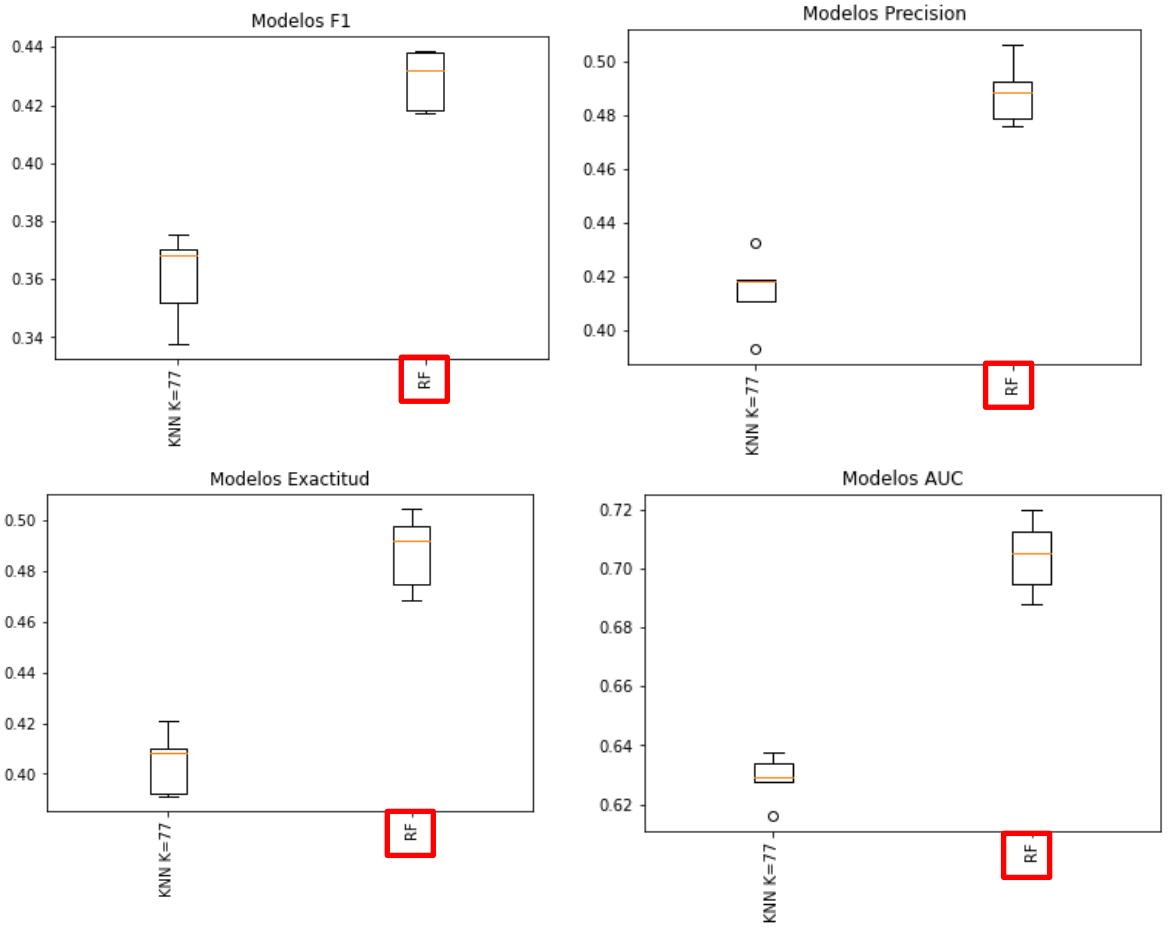
Realizaremos el estudio de cómo afecta variar el número de árboles en nuestro modelo Random Forest:



En la evaluación de los hiperparametros se realizó el estudio de variar los arboles de entrenamiento entre 5 a 100, como vemos en los boxplots de arriba el mejor caso es el de 100 por lo que es el utilizado en el entrenenamiento del modelo.

Radom Forest Vs KNN

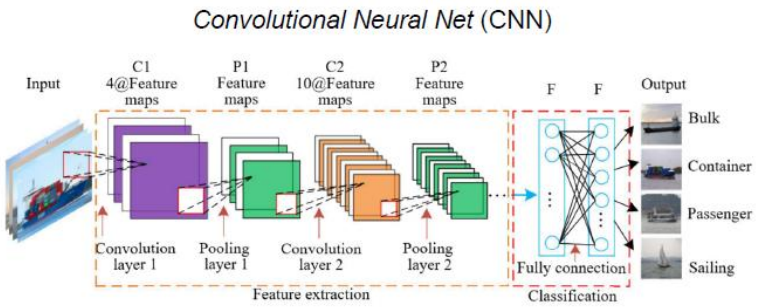
Comparamos los resultados con los obtenidos en el KNN para K=77



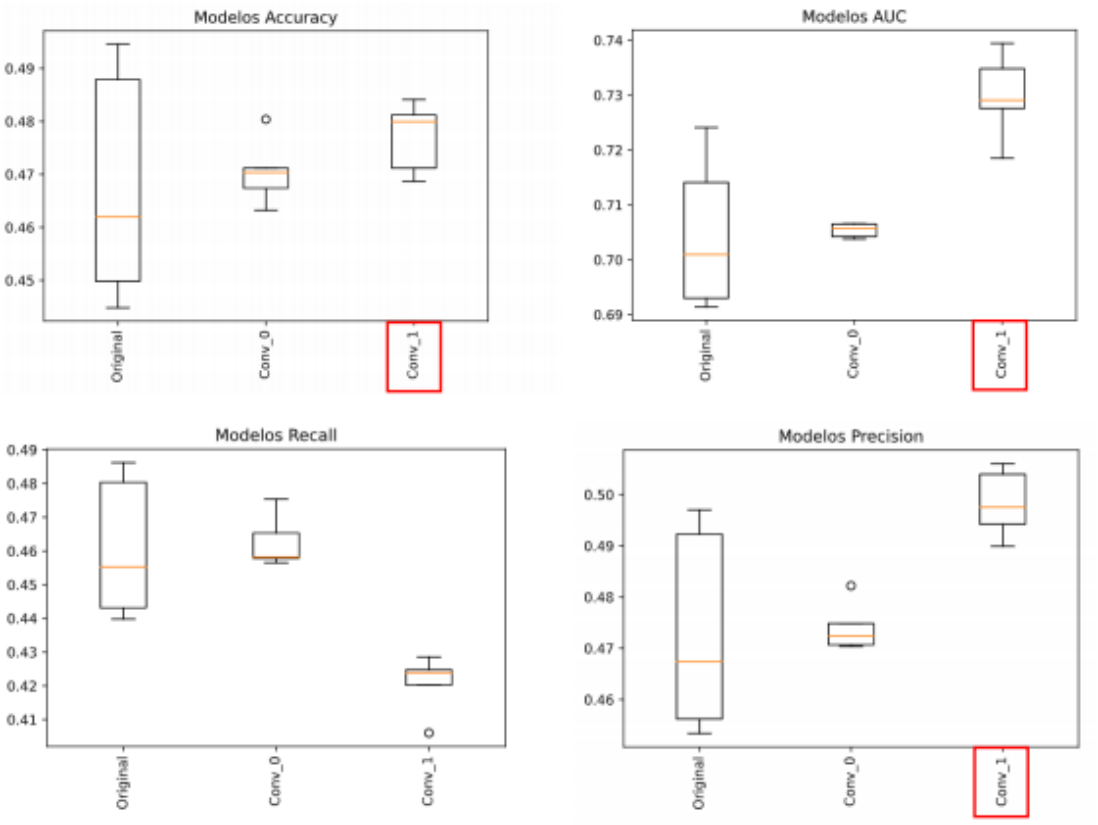
Como observamos en los boxplot, el modelo entrenado como radom forest tiene mejores resultados en cuanto los parámetros de evaluación de la validación cruzada.

Redes Neuronales: Convolución

Modelo de red profunda que es capaz de capturar con éxito las dependencias espaciales y temporales en una imagen mediante la aplicación de filtros relevantes.
La arquitectura se adapta mejor a los datos de imágenes debido a la reducción en el número de parámetros involucrados y la reutilización de pesos.



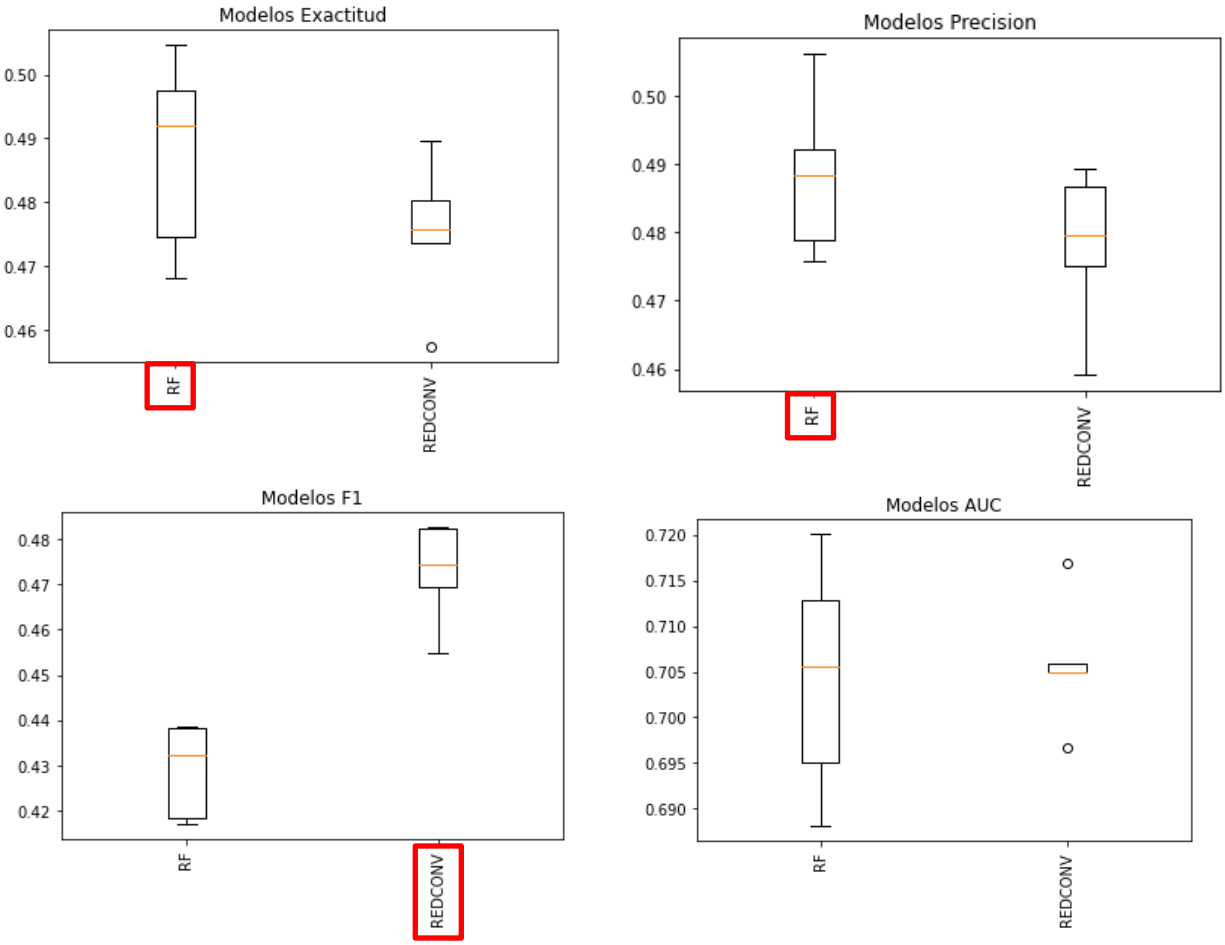
Estudio de Hiperparametros:



Como podemos observar en las gráficas, los valores que pueden mejorar los resultados son los obtenidos en Conv_1, por ser la combinación que arrojó mayor precisión 0,506, tiene 4 capas de neuronas convolucionales de tamaño [8, 8, 16, 16] y dos capas de neuronas densas de tamaño [64, 4].

Red Neuronal Vs Radom Forest

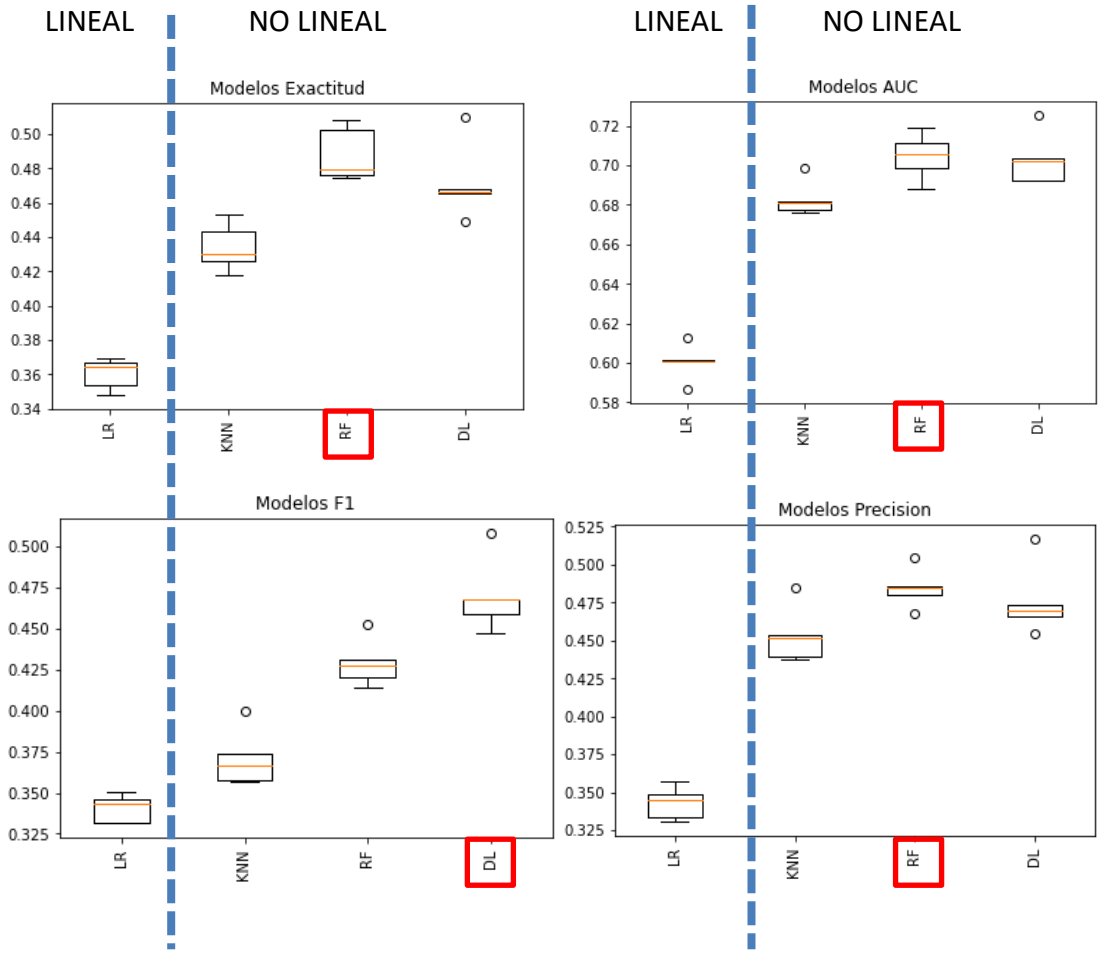
Comparativa del uso de Red neuronal convolucional respecto al uso de Random Forest.



Como observamos en los boxplots la red neuronal respecto al radom forest no representa una mejora en cuanto a los parámetros de evaluación de la validación cruzada.

Conclusiones obtenidas

Como se pueden apreciar en los siguientes Boxplots:



Los métodos de aprendizaje no lineales como KNN, Radom Forest o la Red Neuronal Convolucional al permitir crear regiones de decisión complejas para separar los datos de diferentes clases aportan mejores resultados que los métodos lineales como la regresión logística.

Aunque ningún modelo tiene un resultado significativamente bueno, el modelo escogido entre los cuatro modelos comparados es el Radom Forest debido a que es el que tiene los mejores resultados frente al tiempo de entrenamiento del modelo.