INFORME

Clasificador de imágenes

(Perros y gatos)

Eloy Urriens Arpal

Iker Labairu Lusarreta

11 de diciembre de 2025

Índice

1. Introducción

El objetivo de este informe es redactar la creación y la explotación de varios clasificadores que sean capaces de distinguir imágenes de perros y gatos. Para ello, en los siguientes puntos, se explicará la estructura de un clasificador, los experimentos realizados para observar los resultados, y la conclusión obtenida.

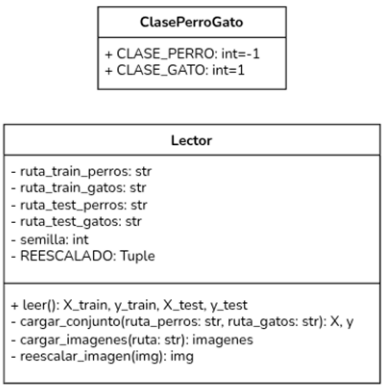
2. Sistema

Para crear el clasificador de imágenes de perros y gatos, se ha optado por diseñar 5 subsistemas. El primero de ellos es el subsistema de lectura, que lee las imágenes y las reescala, después está el de preprocesamiento, que es el encargado de quitar el ruido a las imágenes, luego el transformador, que pasa las características de una imagen a un vector, posteriormente el predictor, que coge el vector y lo clasifica como perro o gato, y finalmente, uno extra para poder experimentar, el experimentador, que sería realmente el clasificador, ya que es el que usa a los anteriores y realiza el flujo completo, pero por varios motivos se ha decidido que tenga el nombre experimentador.

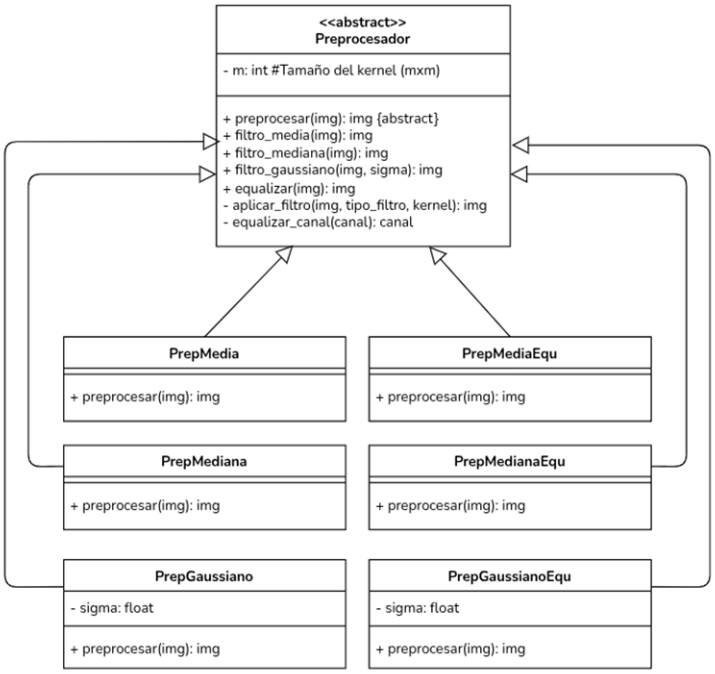
Como información extra, en el código se ha utilizado una librería para compilar código de Python en C, numba. Esto solo se ha utilizado en algoritmos que recorren todos los píxeles de las imágenes, por ejemplo, el extractor de características basado en las texturas y en los preprocesadores. De esta forma, la ejecución de estos era más rápida, sin embargo, en ningún momento se han utilizado más de dos bucles anidados, y además, todas las imágenes cuando las lee el lector las reescala a 256x256 añadiendo padding en negro.

A continuación, se detallan los diagramas de clase de cada subsistema con el fin de facilitar la lectura y corrección del código.

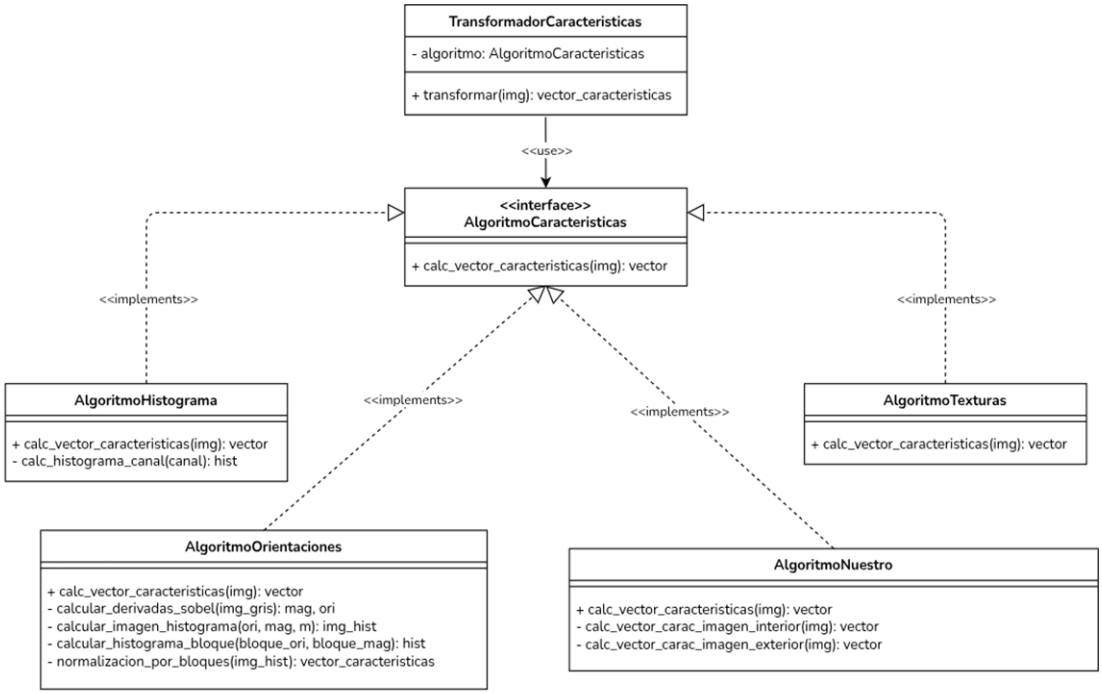
2.1 Subsistema: Lectura



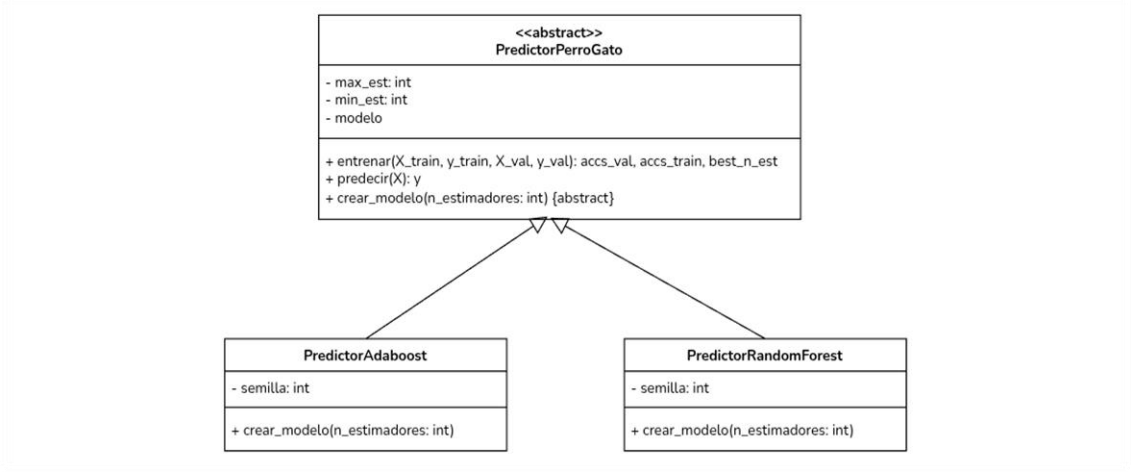
2.2 Subsistema: Preprocesamiento



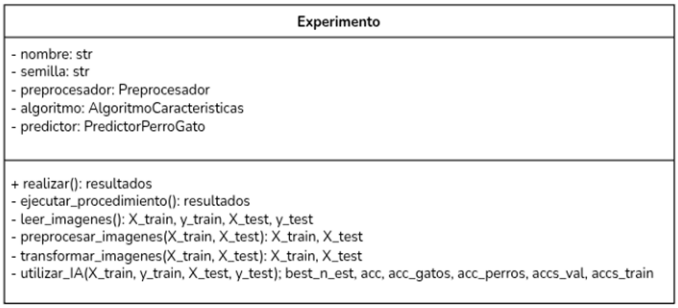
2.3 Subsistema: Transformador



2.4 Subsistema: Predictor



2.5 Subsistema: Experimentador



3. Experimentos

Para evaluar el sistema creado se realizarán 3 fases. En la primera, se generarán múltiples clasificadores para ver los diferentes accuracies, en la segunda, se extraerán de todos los clasificadores creados anteriormente una serie de medidas globales, y en la tercera, se elegirán 4 clasificadores con un mayor accuracy, 1 para cada tipo de algoritmo de extracción de características, y se utilizarán más medidas para extraer más conclusiones.

Sin embargo, antes de continuar, conviene una breve explicación de los datasets que se utilizarán. Estos son 2, de tamaños 200 y 1000 respectivamente, y cuyos ejemplos son imágenes de perros y gatos de diferentes resoluciones. Asimismo, ambas clases están balanceadas, es decir, hay tantas clases de gatos como de perros. Cabe destacar que los tamaños para los conjuntos de datos son del 60% para train, 20% para tests y otro 20% para validación.

3.1 Creación de múltiples clasificadores

Como tenemos 3 subsistemas que componen el núcleo del clasificador (preprocesador, transformador, y predictor), y por cada uno de estos tenemos varios tipos (6 preprocesadores, 4 transformadores y 2 predictores), se ha optado por hacer tantos experimentos como combinaciones de estos, en total 48. Sin embargo, los preprocesadores tienen un tamaño de kernel (mxm), así que se ha escogido m=3 y m=7, por tanto, salen un total de 96 clasificadores.

La configuración general para todos los clasificadores generados viene dada por la configuración de sus preprocesadores, que como ya se ha dicho anteriormente, va a ser de m=3 y m=7, y por la configuración de sus predictores, los cuales son de tipo Adaboost y RandomForest entrenados para ajustarse a los mejores hiperparametros, es decir, el número de estimadores de cada uno entre 1 y 40.

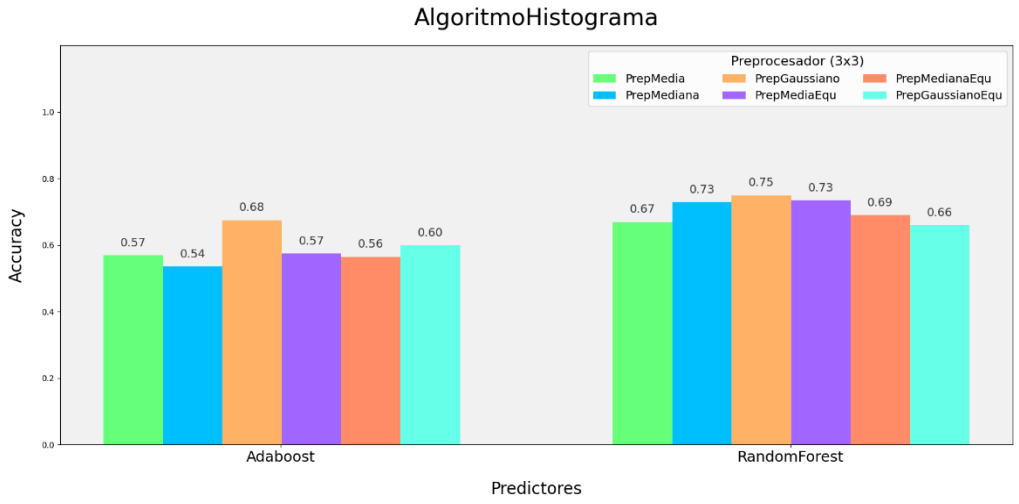
Para recoger todos estos resultados, vamos a divididir la experimentación en 4 partes, de tal forma que en cada una de estas se considera un solo tipo de transfomador, es decir, en la primera parte se generarán todos los clasificadores posibles con el algoritmo del histograma, en la segunda, con el de texturas, en la tercera, con el de orientaciones, y finalmente, con el nuestro.

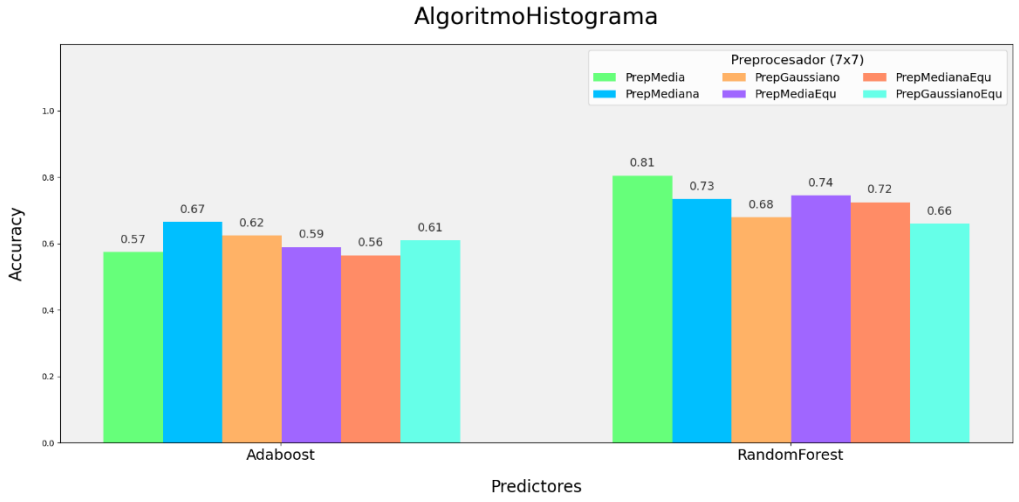
3.1.1 Experimento: Transformador con algoritmo del histograma

En este experimento, tenemos que generar dos grupos de clasificadores, los del tamaño de kernel (3x3), que serían Preprocesadores(6) X AlgoritmoHistograma X Predictores(2)

= 12 clasificadores, y los del kernel (7x7), que sería lo mismo que lo anterior, dando lugar a otros 12 clasificadores.

Para recopilar los anteriores accuracies de cada clasificador se han creado 2 gráficos de barras, uno para cada grupo. A continuación, se pueden ver los resultados obtenidos.





Como se puede observar, el mejor predictor es el RandomForest porque es el que mejores accuracies obtiene, en cualquier caso.

En cuanto al preprocesador, es mucho mejor tener uno de 7x7 que de 3x3, seguramente esto se deba a que como el algoritmo del histograma obtiene el histograma del canal rojo, verde y azul, es muy inestable al ruido de la imagen, por tanto, al utilizar un tamaño de kernel más grande en el preprocesamiento este se disminuye mucho más, asimismo, el tipo de preprocesador que mejor le viene, es el gaussiano de tamaño 3x3 y el de la media en tamaño 7x7.

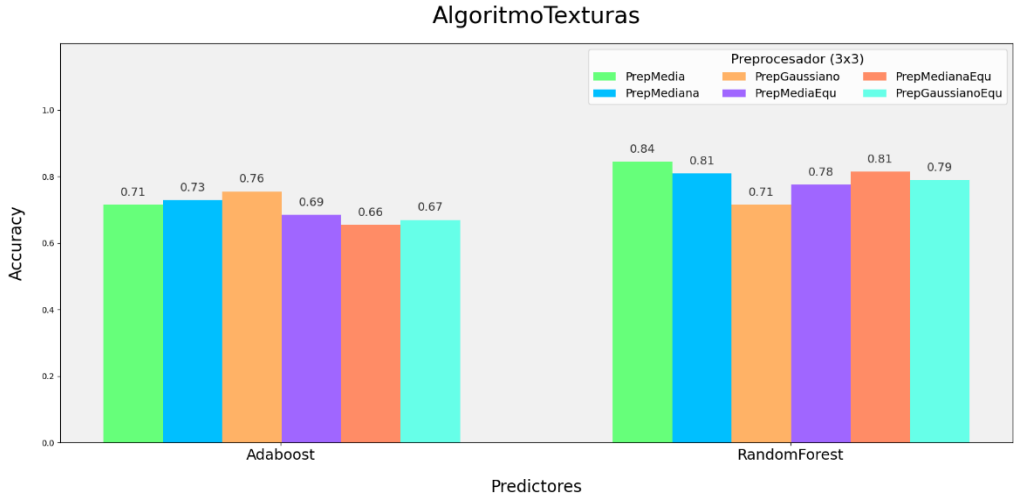
En conclusión, el mejor acc. en test es del 0.81 con PrepMedia(7x7) y RandomForest.

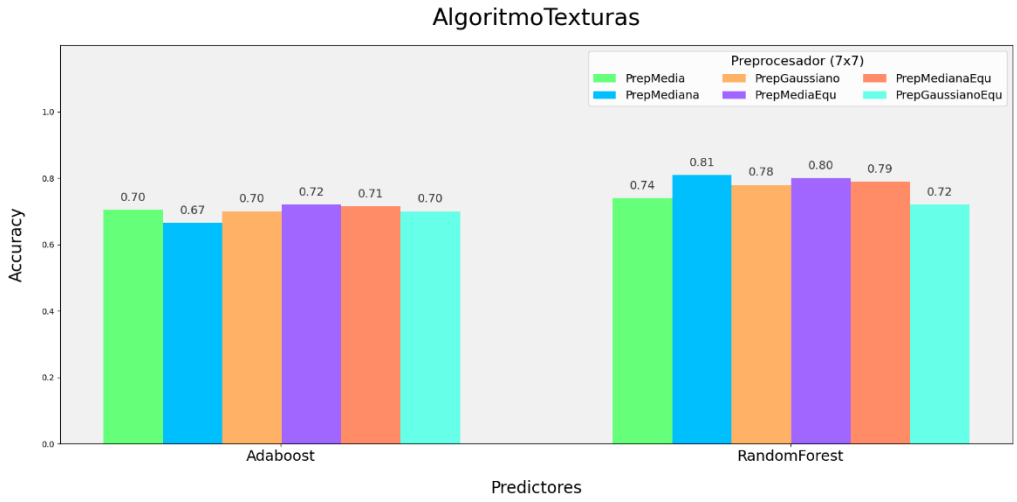
3.1.2 Experimento: Transformador con algoritmo de texturas

En este otro experimento, se volverán a crear los 2 grupos de clasificadores:

- Kernel 3x3: Preprocesadores(6) X AlgoritmoTexturas X Predictores(2) = 12
- Kernel 7x7: Preprocesadores(6) X AlgoritmoTexturas X Predictores(2) = 12

A continuación, se muestran los resultados:





De nuevo, en general, el mejor predictor es el RandomForest, aunque con menor impacto que en el algoritmo del histograma.

Por otro lado, a simple vista, no se puede saber bien que tamaño de kernel funciona mejor en general. Para ello:

Media acc. Adaboost (3x3) = 0.703	Media acc. Adaboost (7x7) = 0.7
Media acc. RandomForest (3x3) = 0.790	Media acc. RandomForest (7x7) = 0.773

Como se puede ver, es mejor un tamaño más pequeño, aunque tampoco hay diferencias significativas. Esto podría ser porque como el algoritmo de texturas trabaja con un tamaño de kernel de 3x3, le viene mejor un filtrado del mismo tamaño en lugar de un filtrado mucho más grande.

Asimismo, para saber que preprocesador es el que mejor le va en general, tampoco se puede ver a simple vista, por ello hacemos los siguientes cálculos:

Media acc. PrepMedia = 0.747;	Media acc. PrepGaussiano = 0.737
Media acc. PrepMediana = 0.755;	Media acc. PrepMediaEqu = 0.747
Media acc. PrepMedianaEqu = 0.742	Media acc. PrepGaussianoEqu = 0.720

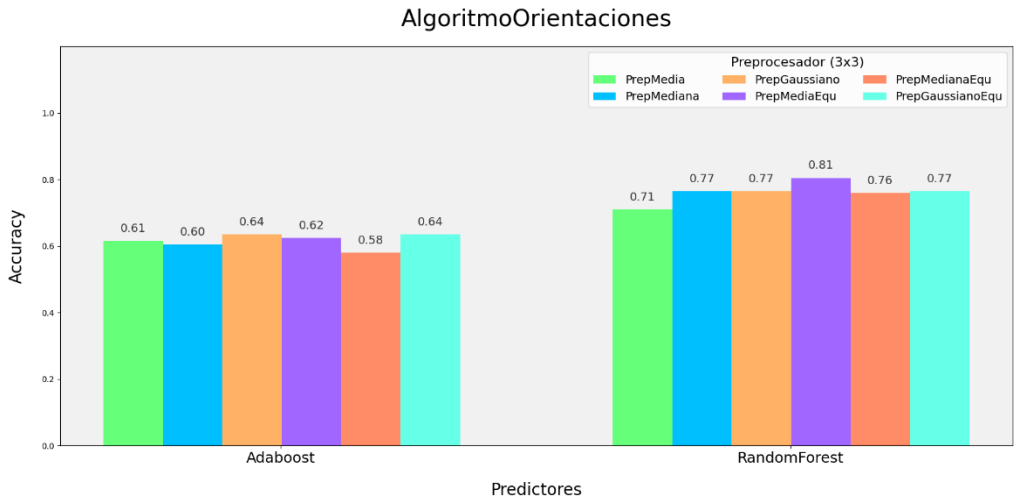
Tampoco hay diferencias significativas, pero el que mejor le viene en general es el de la mediana, seguido del de la media. Esto podría deberse a que al hacer la mediana nunca se inventan datos, es decir, siempre se elige alguna intensidad de la zona común de la imagen con el kernel, sin embargo, al hacer la media o el gaussiano, siempre se hace una suma ponderada que da lugar a un valor que podría no existir en dicha zona, y como el de texturas compara la intensidad del pixel central con los pixeles de alrededor, le va mucho mejor no inventar datos, aunque, se repite, tampoco hay grandes diferencias con el de la media. Por otro lado, también se puede observar que es un poco peor ecualizar la imagen, y de nuevo, podría ser también por la misma razón.

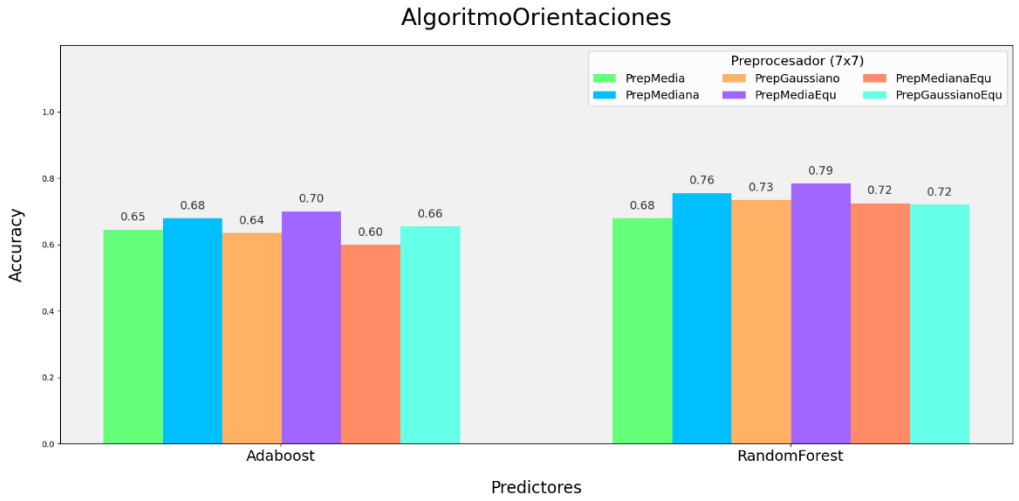
En conclusión, el mejor clasifcador obtenido con el algoritmo de texturas es aquel con el PrepMedia(3x3) y RandomForest, obteniendo un accuracy en test del 0.84.

3.1.3 Experimento: Transformador con algoritmo de orientaciones

En este tercer experimento, de nuevo, se crearán los 2 grupos de clasificadores.

A continuación, se muestran los resultados:





Como se puede ver, en este tercer algoritmo, también resulta mejor el RandomForest en lugar del Adaboost.

En cuanto al preprocesador y su tamaño, en el Adaboost si que se puede ver que en general es mejor uno de 7x7 en lugar de 3x3, pero en el RandomForest ocurre justo lo contrario. La razón de esto podría ser que el Adaboost es más inestable al ruido, y por tanto, funciona mejor con un kernel más grande, y por otra parte, el RandomForest puede ser que sea bastante estable al ruido, y por ello, al aplicar un tamaño de kernel más grande, cosa que no necesitaría, se pierde demasiada información.

Por otra parte, el mejor tipo de preprocesador también puede verse a simple vista, en este caso, sería el de la media con la ecualización, porque reduce el ruido y redistribuye las intensidades para ver detalles más ocultos.

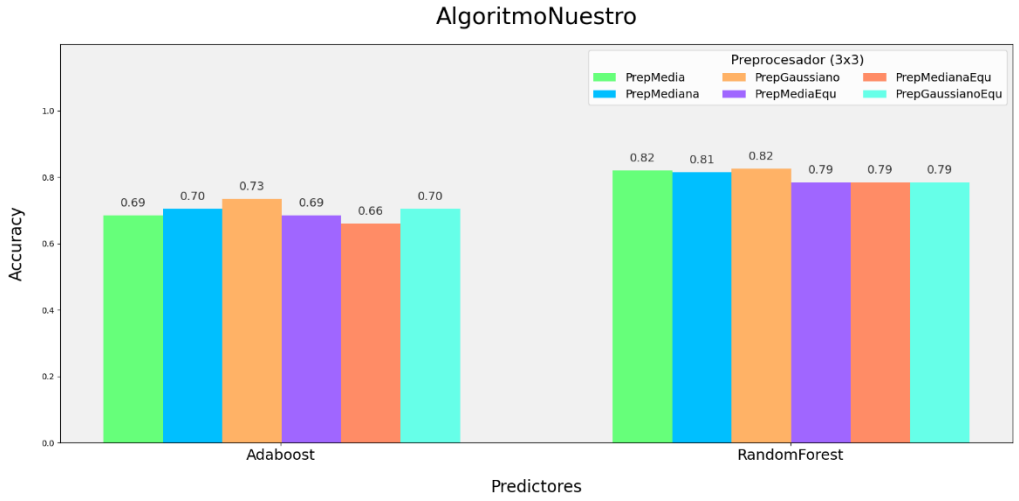
En conclusión, el mejor clasificador generado con el algoritmo de orientaciones se obtiene usando el PrepMediaEqu (3x3) junto con un RandomForest, dando lugar a un accuracy del 0.81.

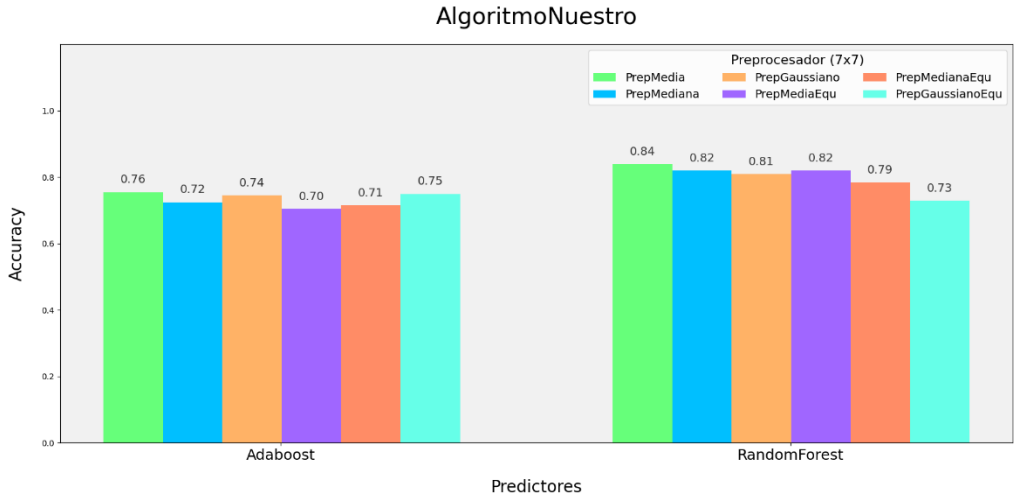
3.1.4 Experimento: Transformador con nuestro algoritmo

Antes de mostrar los resultados, conviene saber en qué consiste nuestro algoritmo. Para ello, supongamos que llega una imagen de 256x256, este algoritmo divide esta imagen en otras 2 imágenes, una imagen interior, y otra exterior. La imagen interior es la zona central de la imagen de tamaño 128x128, y la imagen exterior, el resto de los “bordes” que quedan, es decir, el borde superior de tamaño 256x64, el inferior, del mismo tamaño, el izquierdo, de 64x128 y el derecho, también del mismo tamaño. Una vez extraídas estas dos imágenes, a ambas se le aplican el algoritmo de texturas dando lugar a dos vectores de características de 256 componentes, por tanto, se hace la media ponderada entre estos dos vectores para obtener otro de 256 de tal forma que a la imagen central

se le pone un peso de 0.8 y a la exterior de 0.2, ya que generalmente la información más importante de las imágenes suele estar por el centro en lugar de por los bordes.

Una vez explicado lo anterior, se detallan a continuación los resultados obtenidos.





En estos resultados, al igual que en todos los anteriores, resulta mucho mejor un RandomForest en lugar de un Adaboost.

Asimismo, el tamaño del kernel para el preprocesamiento es mejor de 7x7 en lugar de 3x3, esto parece extraño, porque este algoritmo está construido a partir del de texturas, y en este, era mejor el de 3x3, sin embargo, no tenía había una diferencia significativa entre ambos y puede ser que ahora sí resulte mucho más útil uno de 7x7.

Además, los mejores preprocesadores serían el gaussiano para kernel 3x3 y el de la media para 7x7.

En conclusión, el mejor clasificador que se puede obtener con el algoritmo de texturas es el compuesto por PrepMedia(7x7) y un RandomForest entre 1 y 40 estimadores.

3.2 Medidas globales

Para evaluar los resultados globales de los 96 clasificadores para ver que tal se comportan en general, a continuación, se muestran cuatro medidas calculadas.

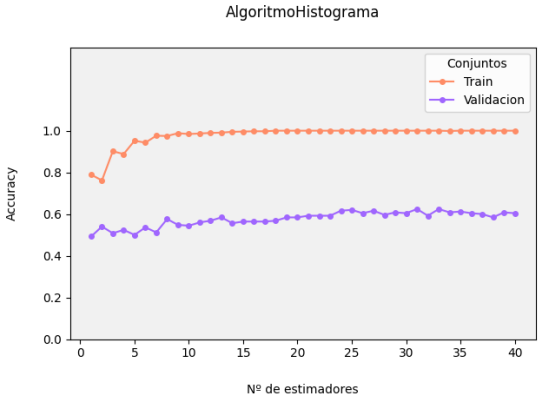
Accuracy	Accuracy Gato	Accuracy Perro	Nº de estimadores
0.71	0.60	0.82	24

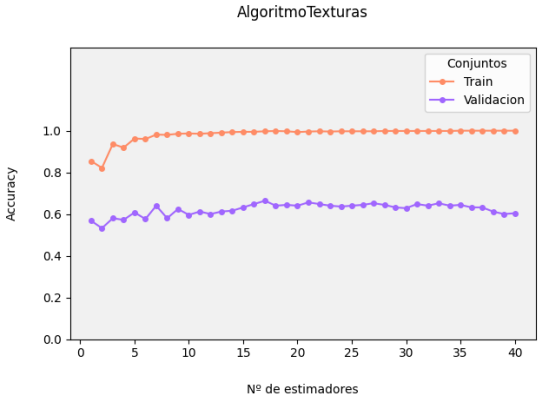
Como se puede ver, en general, los clasificadores obtienen un accuracy del 0.71, esto sería insuficiente si queremos clasificadores precisos, para lograrlos quizás habría que utilizar redes neuronales o mejores algoritmos de extracción de características. Asimismo, los clasificadores predicen mucho mejor a los perros que a los gatos, el accuracy en los perros esta bien, pero el de los gatos es mediocre, no es suficiente. En cuanto al número de estimadores promedio, por clasificador obtenemos unos 24, son pocos, así que por este aspecto para los resultados obtenidos está bastante bien.

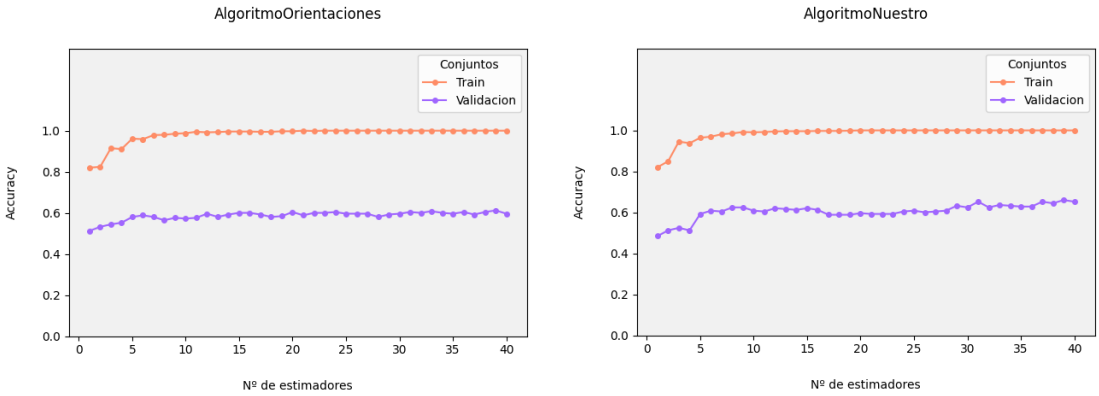
3.3 Mejores clasificadores

En esta sección, se escogen cuatro clasificadores, el que ha tenido mejor rendimiento con el algoritmo del histograma, con el algoritmo de texturas, con el de orientaciones y con el nuestro. Para cada uno de ellos se exponen más medidas tomadas y las curvas de ajuste para los mejores hiperparametros en train y validación.

Algoritmo	Accuracy	Accuracy Gatos	Accuracy Perros	Nº est	Tam. vector	Tiempo
Histograma	0.81	0.70	0.91	31	768	28.38s
Texturas	0.84	0.74	0.95	17	256	23.61s
Orientaciones	0.81	0.69	0.92	39	8100	61.73s
Nuestro	0.84	0.74	0.94	39	256	33.4s







En general, vemos que los 4 son bastante parecidos en cuanto a los accuracies y las curvas de ajuste de hiperparametros, sin embargo, cada uno usan diferentes tamaños de vectores de características, así que por la misma precisión podríamos escoger los que menor tamaño de vector de características tengan. De esta forma, nos aseguramos que los modelos sean también eficientes, por ello, nos quedaríamos con al algoritmo de texturas o con el nuestro.

4. Conclusión

Como se ha visto en la sección 2. Sistema, se han creado varios subsistemas para que al combinarlos se obtengan diferentes clasificadores, los subsistemas principales eran el de preprocesamiento, el extractor de características y el predictor.

Posteriormente, se han ido variando estos para generar varios clasificadores y extraer resultados. El resumen de estos es que el mejor preprocesador es el de la media con tamaño de kernel 7x7, el mejor predictor es el random forest y el mejor extractor de características es el nuestro, porque obtiene en general unos accuracies bastante altos.

Finalmente, en general, los clasificadores creados no son super precisos, ya que para ello habría que usar RRNN o mejores algoritmos de extracción de características, al mismo tiempo, no predicen muy bien a los gatos, pero sí a los perros.