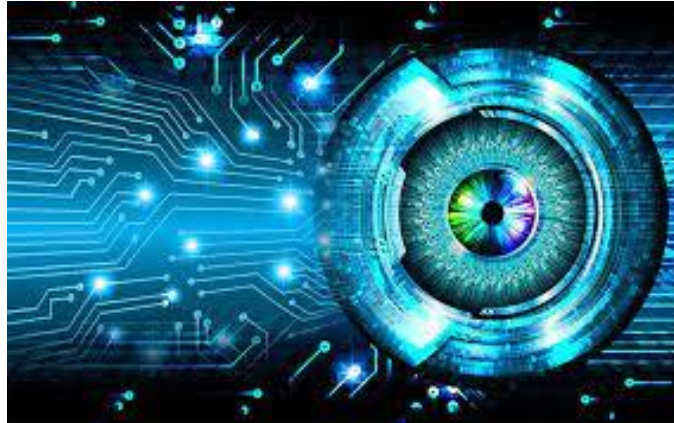


INFORME

VISIÓN ARTIFICIAL



VS



Daniel Pérez Campo e Iván Sola Delgado

ÍNDICE

| | |
|---|---|
| 1.Introducción | 3 |
| 2.Dimensión de las imágenes..... | 3 |
| 3.Extracción de características para las imágenes | 3 |
| 4.Almacenamiento de datasets..... | 3 |
| 5.Resultados obtenidos..... | 4 |
| 5.1. Extracción de características 1 | 4 |
| 5.1.1. Modelo SVM..... | 4 |
| 5.1.2. Modelo regresión logística | 6 |
| 5.1.3. Modelo redes neuronales | 6 |
| 5.1.4. Modelo Adaboost con decisión stump | 6 |
| 5.2. Extracción de características 2 | 7 |
| 5.2.1. Modelo SVM..... | 7 |
| 5.2.2. Modelo regresión logística | 8 |
| 5.1.3. Modelo redes neuronales | 9 |
| 5.1.4. Modelo Adaboost con decisión stump | 9 |
| 5.Conclusión resultados obtenidos | 9 |

1.Introducción

Nuestro problema se basa en el típico problema de perros vs gatos, en el que tenemos que clasificar si hay un perro o un gato en una imagen.

Para realizar este trabajo hemos realizado dos tipos de extracción de características para el conjunto de imágenes con las que trabajaremos.

El objetivo de este trabajo es entrenar un modelo que consiga clasificar las imágenes con un porcentaje de acierto considerable a partir de las características obtenidas.

2.Dimensión de las imágenes

Para entrenar los modelos, los ejemplos que usamos como Xtrain deberán tener el mismo número de características. Por ello hemos implementado una función en la que inserto todas las imágenes y calculo la dimensión media de todas estas.

Esta dimensión que nos devuelve la función nos servirá como referencia para que todas las imágenes sean de esta misma y así, hacer que todas las imágenes tengan el mismo número de características eligiendo una dimensión que sea la media de todas estas, y, por tanto, modificando lo mínimo posible las imágenes originales.





En nuestro caso, la dimensión de nuestras imágenes será de [361, 410], que finalmente tras hacer la extracción será una matriz de [368, 416], para que consigamos un número de bloques justo $(368 \times 416) / (16 \times 16) = 598$ bloques.

















3.Extracción de características para las imágenes

Para la extracción de características hemos considerado los dos tipos de extracción que proporciona la plantilla del trabajo final de la asignatura.

4.Almacenamiento de datasets

Para cada carpeta de imágenes hemos realizado un fichero csv y txt que corresponderá a las características de cada imagen, para así después cargar las características más fácilmente.

| | | | |
|---|----------------|------------------|---------------------|
|  | datasets_1_csv | 07/01/2023 12:49 | Carpeta de archivos |
|  | datasets_1_txt | 07/01/2023 12:52 | Carpeta de archivos |
|  | datasets_2_csv | 07/01/2023 12:53 | Carpeta de archivos |
|  | datasets_2_txt | 07/01/2023 12:53 | Carpeta de archivos |

| | | | | | | | | | |
|---|-------------------|-----------------|-----------------------|------------|---|-------------------|-----------------|---------------------|------------|
|  | cat_100_test.csv | 07/01/2023 4:30 | Archivo de valores... | 6.046 KB |  | cat_100_test.txt | 07/01/2023 4:30 | Documento de tex... | 6.046 KB |
|  | cat_100_train.csv | 07/01/2023 3:26 | Archivo de valores... | 30.236 KB |  | cat_100_train.txt | 07/01/2023 3:26 | Documento de tex... | 30.236 KB |
|  | cat_500_test.csv | 07/01/2023 4:35 | Archivo de valores... | 30.231 KB |  | cat_500_test.txt | 07/01/2023 4:35 | Documento de tex... | 30.231 KB |
|  | cat_500_train.csv | 07/01/2023 4:07 | Archivo de valores... | 151.188 KB |  | cat_500_train.txt | 07/01/2023 4:07 | Documento de tex... | 151.188 KB |
|  | dog_100_test.csv | 07/01/2023 4:31 | Archivo de valores... | 6.043 KB |  | dog_100_test.txt | 07/01/2023 4:31 | Documento de tex... | 6.043 KB |
|  | dog_100_train.csv | 07/01/2023 3:33 | Archivo de valores... | 30.205 KB |  | dog_100_train.txt | 07/01/2023 3:33 | Documento de tex... | 30.205 KB |
|  | dog_500_test.csv | 07/01/2023 4:38 | Archivo de valores... | 30.205 KB |  | dog_500_test.txt | 07/01/2023 4:38 | Documento de tex... | 30.205 KB |
|  | dog_500_train.csv | 07/01/2023 4:30 | Archivo de valores... | 151.067 KB |  | dog_500_train.txt | 07/01/2023 4:30 | Documento de tex... | 151.067 KB |

5.Resultados obtenidos

5.1. Extracción de características 1

Disponemos de imágenes con un tamaño final de 368x416 y un total de 598 bloques con 256 características cada una, es decir cada imagen esta formado por $598 \times 256 = 153.088$ características.

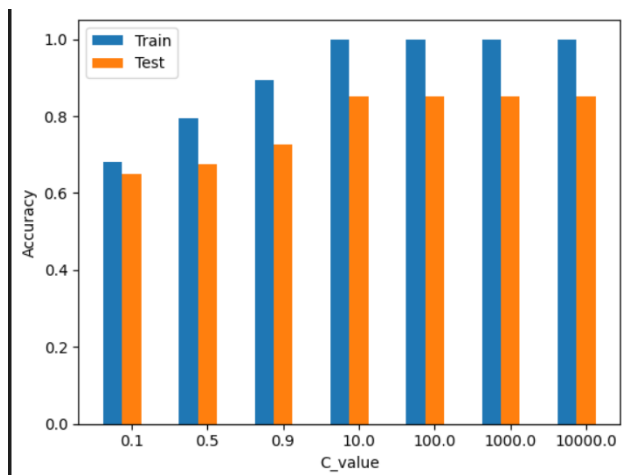
5.1.1. Modelo SVM

Para este modelo primero haremos comprobaciones con el modelo sin kernel, ya que según la teoría en los casos en el que el número de características es mucho mayor que el número de ejemplos es mejor implementar SVM sin kernel.

Pruebas con 200 imágenes en train y 40 en test:

Para SVM sin hiperparámetros, obtenemos un 75% de precisión en test y un 93.5 % en entrenamiento.

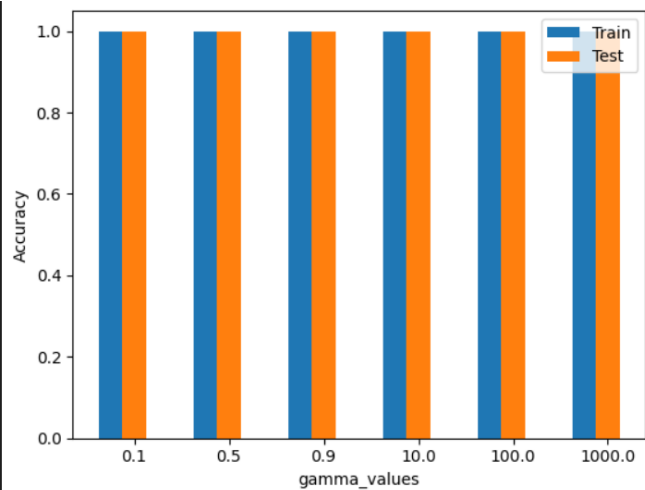
Para SVM con hiperparámetros, usando únicamente el valor de la constante C, obtenemos las siguientes precisiones en base al valor de C.



Podemos observar que a partir de $C=10.0$, ya no mejora, quedándonos con la mejor precisión en test de 85% y en train del 100%.

Para SVM con hiperparámetros, usando el valor de la constante $C=10$, con este valor de C ya conseguimos la mejor precisión obtenida. Ahora añadiremos también la constante de gamma, y por tanto utilizaremos SVMs con kernel.

En este caso, como observamos en la gráfica de barras, usando la constante gamma, en cualquier situación obtenemos un 100% de precisión.

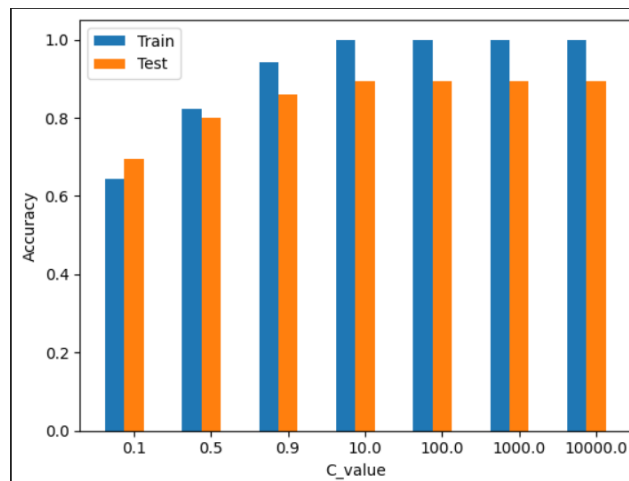


Pruebas con 1000 imágenes en train y 200 en test:

Realizaremos las mismas pruebas realizadas que en el anterior caso.

Con SVM sin parámetros obtenemos una precisión de 86.5% en test y 95% en entrenamiento.

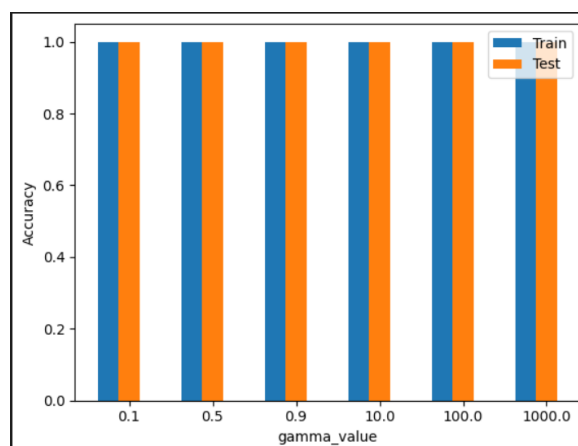
Usando C como hiperparámetro obtenemos la siguiente gráfica de barras para cada valor de C:



Al igual que en el anterior caso, a partir de 10.0 la precisión no mejora.

Obtenemos una máxima de 1 en train y 89.5% en test y 100% en train

Para SVM con hiperparámetros, usando el valor de la constante C=10, con este valor de C ya coseguimos la mejor precisión obtenida. Ahora añadiremos también la constante de gamma, y por tanto utilizaremos SVMs con kernel.



Al igual que en el anterior caso obtenemos un 100% de precisión en todos los casos.

En conclusión, los modelos SVMs con kernel se adaptan mejor a las características extraídas que los SVM sin kernel, aunque en ambas obtenemos precisiones más que considerables.

5.1.2. Modelo regresión logística

Pruebas con 200 imágenes en train y 40 en test:

Obtenemos una precisión de 85% en test y 100% en train.

Pruebas con 1000 imágenes en train y 200 en test:

Obtenemos una precisión de 85% en test y 100% en train.

Hemos obtenido la misma precisión, en ambas situaciones. Curioso.

5.1.3. Modelo redes neuronales

Pruebas con 200 imágenes en train y 40 en test:

Obtenemos una precisión de 50% en test y 50% en train.

Pruebas con 1000 imágenes en train y 200 en test:

Obtenemos una precisión de 50% en test y 50% en train.

Es un modelo malo para clasificar en nuestro problema.

5.1.4. Modelo Adaboost con decisión stump

Este modelo es considerado uno de los mejores en datamining. Veremos si será eficiente en nuestro problema de clasificación de imágenes. Utilizaremos un número de 30 clasificadores con una profundidad de 1 para decisión stump.

Pruebas con 200 imágenes en train y 40 en test:

Obtenemos una precisión de 82.5% en test y 100% en train.

Pruebas con 1000 imágenes en train y 200 en test:

Obtenemos una precisión de 72.5% en test y 86.6% en train.

5.2. Extracción de características 2

Disponemos de imágenes con un tamaño final de 368x416 y un total de 598 bloques con 9 características cada una, es decir cada imagen está formado por $598 \times 9 = 5.382$ características.

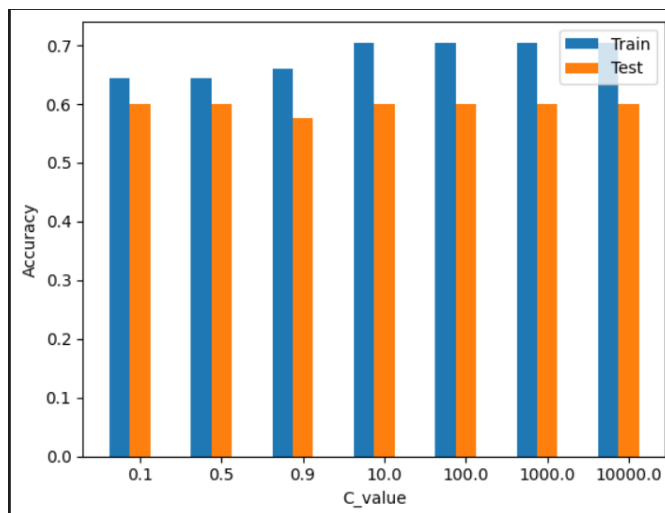
5.2.1. Modelo SVM

Para este modelo, en este caso como el número de características se asemeja al número de ejemplos que tenemos usaremos tanto SVM con kernel y sin kernel.

Pruebas con 200 imágenes en train y 40 en test:

Para SVM sin hiperparámetros, obtenemos un 59.5% de precisión en test y un 64.0 % en entrenamiento.

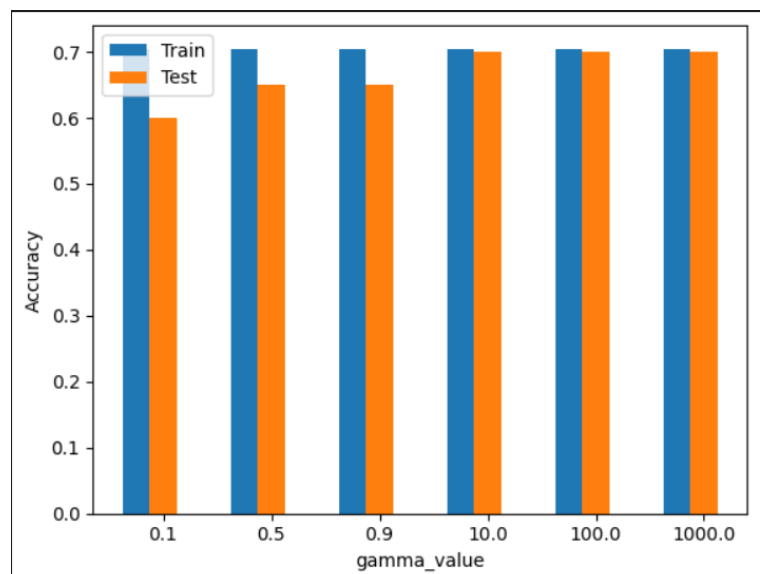
Para SVM con hiperparámetros, usando únicamente el valor de la constante C, obtenemos las siguientes precisiones en base al valor de C.



Podemos observar que a partir con $C=0.1, 0.5, 10.0, 100.0, 1000.0$, obtenemos una precisión en test de 60% y con $C > 10$ 70.5% en train.

Para SVM con hiperparámetros, usando el valor de la constante $C=10$, con este valor de C ya conseguimos la mejor precisión obtenida. Ahora añadiremos también la constante de gamma, y por tanto utilizaremos SVMs con kernel.

En este caso, como observamos en la gráfica de barras, usando la constante gamma, en cualquier situación obtenemos un 70.5% de precisión en entrenamiento y un 70% de precisión en test con gamma igual a 10 o mayor.

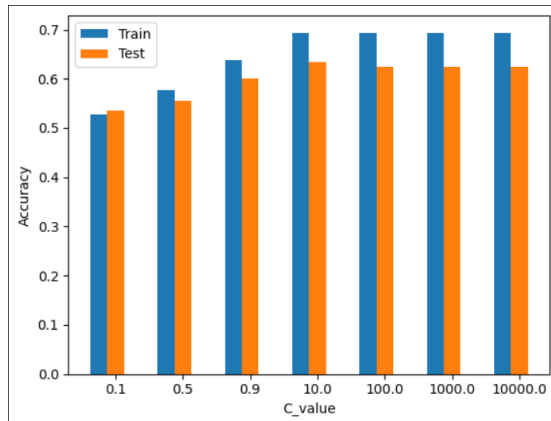


Pruebas con 1000 imágenes en train y 200 en test:

Realizaremos las mismas pruebas realizadas que en el anterior caso.

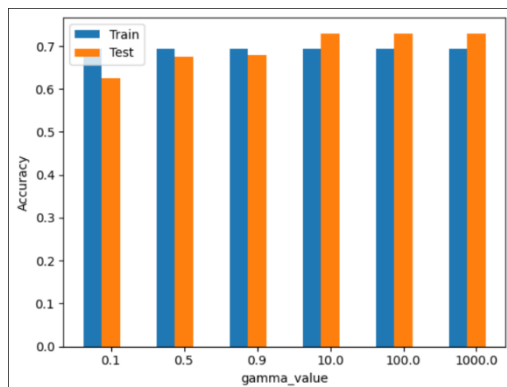
Con SVM sin parámetros obtenemos una precisión de 57.5% en test y 66.0% en entrenamiento.

Usando C como hiperparámetro obtenemos la siguiente gráfica de barras para cada valor de C:



En este caso, a partir de $C=10.0$ la precisión no mejora. Obtenemos un 69.4% en train y 63.5% en test.

Para SVM con hiperparámetros, usando el valor de la constante $C=10$, con este valor de C ya coseguimos la mejor precisión obtenida anterior. Ahora añadiremos también la constante de gamma, y por tanto utilizaremos SVMs con kernel.



Con un valor de $C = 10$ y $\gamma = 10$ o más obtenemos un 73% de aciertos en test, en train obtenemos siempre un 69.4% sea el valor que sea gamma.

En conclusión, esta extracción de características en los modelos SVM funcionan peor, podría ser porque el número de características que tenemos es mucho menor, aún así conseguimos obtener un 70% de aciertos en test. La ventaja que podríamos sacar de esta extracción en comparación a la otra podría ser que como disponemos de menos características el tiempo de ejecución del modelo es mucho menor.

5.2.2. Modelo regresión logística

Pruebas con 200 imágenes en train y 40 en test:

Obtenemos una precisión de 60.0% en test y 70.5% en train.

Pruebas con 1000 imágenes en train y 200 en test:

Obtenemos una precisión de 62.0% en test y 68.8% en train.

No mejora con respecto a las SVM.

5.1.3. Modelo redes neuronales

Pruebas con 200 imágenes en train y 40 en test:

Obtenemos una precisión de 62.5% en test y 70.5% en train.

Pruebas con 1000 imágenes en train y 200 en test:

Obtenemos una precisión de 61.0% en test y 64.0% en train.

Es un modelo malo para clasificar en nuestro problema.

5.1.4. Modelo Adaboost con decisión stump

Este modelo es considerado uno de los mejores en datamining. Veremos si será eficiente en nuestro problema de clasificación de imágenes. Utilizaremos un número de 30 clasificadores con una profundidad de 1 para decisión stump.

Pruebas con 200 imágenes en train y 40 en test:

Obtenemos una precisión de 57.5% en test y 70.0% en train.

Pruebas con 1000 imágenes en train y 200 en test:

Obtenemos una precisión de 53.0% en test y 59.5% en train.

5.Conclusión resultados obtenidos

Resumiendo, nuestro trabajo, hemos realizado pruebas con 4 modelos, los cuales se comportan de manera distinta a la hora de clasificar los datos. ¿Cuál de todos los modelos utilizados es el que más se podría adaptar a nuestro problema?

En nuestro caso los mejores resultados han sido obtenidos con el modelo SVM con kernel. Llegando a un 100% de precisión en el primer tipo de extracción de características y a un 73% como máximo en el segundo. Hemos necesitado sobrentrenar el modelo para obtener una buena precisión.

Cabe destacar que el segundo tipo de extracción, no podemos consolidar que sea una buena forma de extraer las características para nuestro problema. El mejor caso que hemos obtenido es un 73%. Teniendo en cuenta que es un problema de clasificación binaria podríamos considerar esta, una mala precisión.

