Modelos y Simulación II: Proyecto del curso

Daniel Santa Rendón, Daniel Torres González

Ingeniería de sistemas, Universidad de Antioquia
Medellín, Colombia

daniel.santar@udea.edu.co
daniel.torresg@udea.edu.co

***Resumen*—*Segment es una base de datos proporcionada por UCI repository que está compuesta de muestras a partir de imágenes segmentadas de 7 imágenes exteriores. En este trabajo, se procede a trabajar con el dataset ya mencionado. Para realizar este aprendizaje supervisado, primeramente, se hace un pre-procesamiento de los datos, en este caso, únicamente, se cambiaron el nombre de las salidas que eran categóricas a numéricas, dándole a cada un valor entero. Después, se ejecutaron los modelos, los cuales, se les varían los parámetros buscando cuál tiene los mejores resultados, por ejemplo, el número de vecinos en KNN, las neuronas por capa en perceptrón multicapa, árboles en random forest, entre otros parámetros de los demás modelos ejecutados. Una vez que cada modelo fue ejecutado, se seleccionan los resultados más óptimos junto con los parámetros que permiten obtener dichos resultados. Luego, se realiza el análisis de Fisher, cuyo objetivo es saber qué variables tienen más y menos discriminación, estas últimas pueden ser candidatas a eliminar. Con el ánimo de comparar entre sí el rendimiento y el comportamiento posterior a la ejecución de los modelos, se hace una selección de características descendente, además se realiza una extracción de características con PCA. Por último, se compara el resultado de cada uno de los modelos sin ninguna variación en sus características con los resultados obtenidos en la selección y luego se compara nuevamente la base de datos completa con la extracción PCA.*

Palabras claves—Segmentación de imágenes, machine learning, aprendizaje supervisado, clasificación.

Link -
http://archive.ics.uci.edu/ml/datasets/Image+Segmentation

Introducción:

Una de las necesidades que la inteligencia artificial puede resolver es la clasificación de datos por medio de algoritmos, que, utilizando un modelo puede dar resultados satisfactorios a la agrupación automática de un conjunto de muestras. En este trabajo, se tiene una base de datos extraída a partir de imágenes de exteriores segmentadas y están clasificadas en siete grupos distintos. Con la información obtenida de la base de datos se tiene como propósito realizar un aprendizaje de máquina supervisado con los siguientes modelos de clasificación: Na¨ıve Bayes, K vecinos más cercanos, Redes Neuronales Artificiales, Random Forest y Máquinas de Soporte Vectorial con kernel lineal y RBF. La intención de utilizar diferentes modelos es reconocer el comportamiento y rendimiento que tiene cada uno de ellos clasificando, en este caso, imágenes segmentadas.

En la sección 1 se hace la descripción de la base de datos, en la sección 2 se describen las variables de entrada y salida de la base de datos, en la sección 3 se hace un repaso de las investigaciones referentes a la base de datos, en la sección 4 se hacen los experimentos con los datos, en esta sección se incluye la metodología, la cual consta del pre procesamiento de datos, los modelos Naïve Bayes, K vecinos más cercanos, Redes neuronales artificiales, random forest y máquinas de soporte vectorial. En esta última sección también se incluye el resultado del índice de Fisher, la selección de características hacia atrás y la extracción de características PCA. Por último, en la sección 5 se hace el análisis de resultados.

1. Descripción de la base de datos:

La base de datos *Image Segmentation* está compuesta por 2310 muestras extraídas aleatoreamentes de 7 bases de datos de imágenes de exteriores. Cada muestra está compuesta por una región de 3 x 3 píxeles, para poder extraer cada muestra, las imágenes fueron segmentadas a mano y clasificadas cada una en alguna de las 7 diferentes clases que son: *brickface* (ladrillo), *sky* (cielo), *foliage* (follage), *cement* (cemento), *window* (ventana), *path* (camino) y *grass* (hierba). Con esta base de datos se quiere darle la etiqueta correspondiente una región de 3 x 3 píxeles dependiendo a qué parte del exterior pertenece. La base de datos está dividida en entrenamiento y test, la partición de entrenamiento consta de 210 muestras (30 muestras por clase), mientras que la

partición de test consta de 2100 muestras (300 muestras por clase).

2. Variables de la base de datos:

Con estos datos se requiere entrenar un modelo el cual recibe 19 entradas:

- **Variables de entrada**

0.region-centroid-col: La columna del pixel central de la muestra.
1.region-centroid-row: La fila del pixel central la muestra.
2.region-pixel-count: El número de parámetros en una región, este número es = 9.
3.short-line-density-5: Resultado de un algoritmo de extracción que cuenta cuántas líneas de tamaño 5 con bajo contraste, menor o igual a 5, pasan por la región en cualquier dirección.
5.short-line-density-2: Es igual a short-line-density-5 pero cuenta cuántas líneas de tamaño 5 con alto contraste, mayor que 5, pasan por la región.
5.vedge-mean: Mide el contraste de píxeles adyacentes horizontalmente en la región. Hay 6, se dan la media y la desviación estándar. Este atributo se utiliza como detector de borde vertical.
6. vegde-sd: (ver 6)
7.hedge-mean: Mide el contraste de píxeles verticalmente adyacentes. Utilizado para la detección de línea horizontal.
8. hedge-sd: (ver 8).
9.intensity-mean: El promedio sobre la región de (R + G + B)/3
10.rawred-mean: El promedio sobre la región del valor R.
11.rawblue-mean: El promedio sobre la región del valor B
12.rawgreen-mean: El promedio sobre la región del valor G.
13.exred-mean: Mide el exceso de rojo: (2R - (G + B))
14.exblue-mean: Mide el exceso de azul: (2B - (G + R))
15.exgreen-mean: Mide el exceso de verde (2G - (R + B))
16.value-mean: Transformación no lineal 3-d de RGB. (El algoritmo se puede encontrar en Foley an VanDam, Fundamentals of Interactive Computer Graphics)
17. saturatoin-mean: (ver 17)
18. hue-mean: (ver 17)

- **Variables de salida**

  Este modelo sólo tiene una salida. Esta salida retorna a cuál de las 7 clases pertenece la muestra.

El tipo de codificación de todos los atributos de la base de datos es contínua.

La base de datos no cuenta con ningún valor faltante, lo cual, hace innecesario aplicar algún método de llenado.

3. Estado del arte:

Esta base de datos aunque está catalogada como un problema de clasificación, la mayoría de artículos la usan para realizar *clustering*, esto se interpreta de manera lógica puesto que la clasificación de los pixeles fue hecha de manera manual, por lo que encontrar un método de agrupamiento eficiente es un reto interesante. De algunos artículos que ponían en práctica estos métodos no supervisados, analizamos el tratamiento que le hacen a los datos: (Vlassis, N., & Likas, A. 2002) aplican un análisis de componentes principales obteniendo 5 variables que explican más del 95% de la varianza de la base de datos. En (Tung, A. K. H., Xu, X., & Ooi, B. C. 2005) eliminan de entrada las características 5, 7 y 9 pues, afirman, son redundantes con las características 4, 6 y 8; además normalizan los datos en un intervalo de [-5, 5] sin embargo no especifican qué método de normalización usaron; puede ser interesante adoptar esta configuración para algoritmos supervisados.

Por otro lado, (J.T.-Y. Kwok, 1999) implementa un SVM con salidas moderadas, lo que significa que se controla el valor de los pesos **W** asignándoles un rango de valores posibles, con este método obtuvo un error del 8.4% con los datos de prueba (2100 datos) y lo compara con un SVM usando un evidence *framework*, este es un *framework* Bayesiano que consta de tres niveles: 1. Se le da un valor a □ y se infiere la posterior distribución de **W**, 2. Se determina un nuevo valor para □ y 3. se evalúan los diferentes modelos examinando sus probabilidades. Kwok construye un clasificador por clase para un patrón particular y el clasificador con mayor salida se selecciona y se asigna la etiqueta a esta clase. Con este método obtuvo un error del 9.8%.

(Lindenbaum, M. Markovitch, S & Rusakov, D. 1999) proponen un muestreo selectivo utilizando un *random field model* (modelo de campo aleatorio), implementado en KNN. Lo que hace este algoritmo es encontrar la muestra con mayor utilidad considerando su efecto en el clasificador resultante, para esto se requiere estimar la probabilidad de las posibles etiquetas, es decir, es un procedimiento iterativo en el que en cada iteración se llama al muestreo selectivo para obtener una muestra sin clasificar y se llama al maestro o profesor para que etiquete ese dato. El mejor resultado que obtuvieron fue de un error promedio del 10.9% y 1.4% en la desviación estándar. Ellos normalizaron los datos entre [0, 20] y transformaron el problema en un

problema biclase siendo 3 clases como 1 y 4 clases como 4.

(Albert Ribes, 2019) realiza utilizan la base de datos *Image Segmentation* en los algoritmos *Support Vector Machine*(*SVM), Decision Tree(DT), Random Forest(RF)* y *Logistic Regression(LG).* La metodología de validación que utilizó fue cross-validation para cada uno de los modelos, los mejores resultados que obtuvo fueron:

- Para SVM con kernel lineal fue de 0.023 en entrenamiento y 0.042 en validación.
- Para DT el error fue de 0.041.
- Para RF el error fue de 0.028.
- Para LG el error de entrenamiento fue 0.036 y el error de prueba fue 0.048.

Aunque es una tesis de grado, es interesante mencionarla en este trabajo puesto que ofrece un rango más amplio de comparación que los demás artículos.
También es importante aclarar que todos los autores exceptuando a Albert Ribes usan el conjunto de training (210 muestras) y test (2100 muestras) por lo que sus resultados serán menos óptimos.

4. Experimentos

Nombre:		Image Segmentation Data Set			
Características del conjunto de datos:	Multivariable	**Número de muestras:**	2310	**Fecha de donación:**	1990-11-01
Tipo de atributos:	Real	**Número de variables:**	19	**Total de clases:**	7
Tarea asociada:	Clasificación	**Valores faltantes:**	No	**Muestras por clase:**	330
Metodología de validación:		Validación Bootstrapping (75% entrenamiento- 25% validación) con 4 iteraciones			

Figura 1: Descripción de base de datos.

La metodología fue la siguiente:

A. Preprocesamiento

Como la base de datos no tiene ni variables categóricas, ni desbalance, ni datos faltantes el único trabajo de preprocesamiento fue convertir las clases a etiquetas numéricas, esto se realizó desde el archivo .txt con la función de buscar y reemplazar, quedando así: *Brickface* = 0, *Sky* = 1, *foliage* = 2, *cement* = 3, *window* = 4, *path* = 5 y *grass* = *6.* Para la matriz de confusión será: *Brickface* = Clase A, *Sky* = Clase B, *foliage* = Clase C, *cement* = Clase D, *window* = Clase E, *path* = Clase F y *grass* =Clase G.

B. Modelos

En esta fase es donde se entrenan y se validan los modelos de ML (Naive Bayes, K vecinos más cercanos, Perceptrón multicapa, Random Forest y máquinas de soporte vectorial con kernel lineal y kernel rbf). Para cada modelo se usó la validación bootstrapping con 75% de los datos para entrenamiento y 25% de los datos para prueba con 4 iteraciones. Las índices que se utilizaron fueron la eficiencia, la sensibilidad, la precisión, F-score, error de clasificación en la validación, los respectivos intervalos de confianza para cada índice y se midió el tiempo de ejecución.
Todos los modelos fueron ejecutados en Jupyter bajo el lenguaje de programación python, haciendo uso de las librerías que ofrece este lenguaje para el cálculo matemático y para los modelos de ML.
Es importante aclarar que para este trabajo se usó el total de muestras (2310) para el entrenamiento y validación, es decir, no se tiene un subconjunto de muestras usado únicamente para prueba.

- ***Naïve Bayes:***

El clasificador Naive Bayes, es un clasificador fundamentado en el teorema de Bayes. Este algoritmo considera que cada una de las características contribuye de manera independiente a la probabilidad de que una muestra pertenezca a una clase. Este algoritmo crea una tabla de frecuencia, que, a partir de esta se crea una tabla de probabilidad, con esto se calcula la probabilidad para cada clase y la clase con la probabilidad más alta es el resultado de la predicción. Para los resultados completos ver apéndice 1.
El mejor resultado generado por este modelo fue:

Eficiencia	Sensibilidad	Precisión	F-Score	Error_Prueba	Tiempo ejecución(seg)
0.7967	0.7972	0.8153	0.7789	0.2033	0.026

Figura 2. Mejor resultado algoritmo Naïve Bayes.

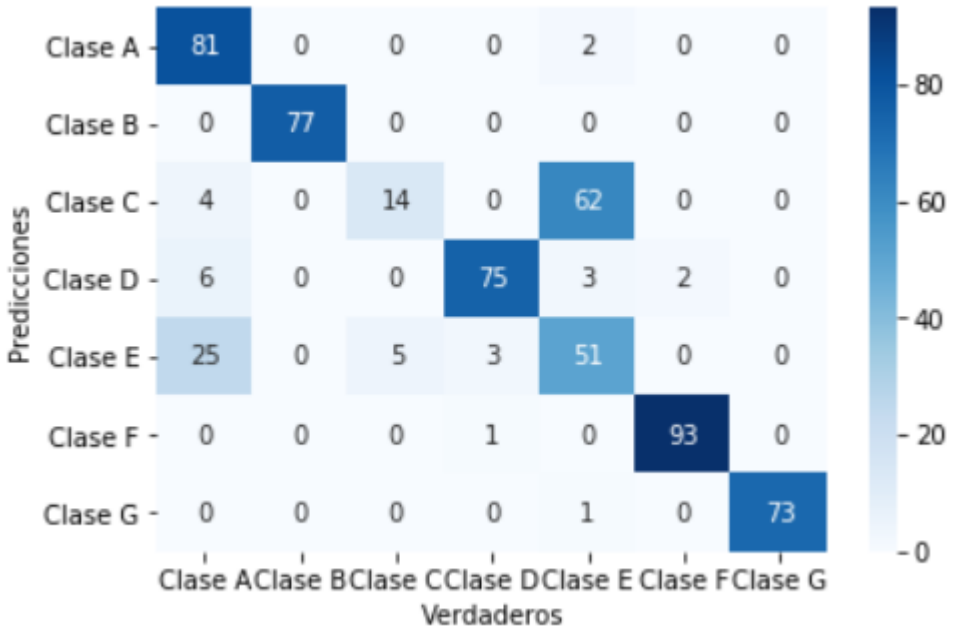

Figura 3.Matriz de confusión del algoritmo Naive Bayes.

Este algoritmo fue el que peores resultados arrojó, el error de clasificación fue de un 20.3% y como se evidencia en la matriz de confusión se clasificaron mal 114 muestras, de las cuales 62 fueron clasificadas como clase C pero que en realidad eran de la clase E. Es interesante ver que este modelo el tiempo de ejecución fue el más bajo de todos pero el que más error tuvo.

- ***K vecinos más cercanos:***

Para este algoritmo, primero se seleccionan 2 parámetros: el número de vecinos y la medida de distancia que se utilizará. Luego, se almacena cada entrada con su respectiva salida □□. Una vez hecho esto, cuando ingrese una nueva muestra, se calcula la distancia entre la muestra ingresada y todas las demás muestras almacenando los valores en un vector. Se buscan las k muestras con los valores de distancia menores y se extrae su valor de salida □□, la clase asignada a la muestra será la clase que más se presente en el conjunto de vecinos más cercanos. En nuestro caso, se ejecutó el algoritmo variando el número de vecinos en el rango [1, 20] y utilizando la distancia euclidiana. Para los resultados completos ver apéndice 2. El mejor resultado obtenido fue:

#vecinos	Eficiencia	Sensibilidad	Precisión	F-Score	Error_Prueba	Tiempo ejecución(seg)
1	0.9658	0.9677	0.9661	0.9663	0.0342	0.0941

Figura 4. Mejor resultado algoritmo K vecinos más cercanos.

Figura 5. Matriz de confusión con 1 vecino.

Este algoritmo presenta una mejora importante al resultado obtenido, un error de 3.4% usando k=1 como parámetro. En la matriz de confusión se puede apreciar que el modelo generalizó bastante bien, pues la clase que peor predijo fue la clase C y la clase E con 5 muestras mal clasificadas en cada una. En este modelo el tiempo de ejecución sigue siendo muy bajo y con resultados bastante buenos, por lo que para este problema es muy conveniente usarlo.

- ***Perceptrón multicapa:***

El perceptrón multicapa es una red neuronal compuesta por un conjunto de unidades más pequeñas llamados perceptrones simples. Esta red neuronal tiene la capacidad de procesar cálculo más complejos a medida que se le agregan más perceptrones haciéndolo un algoritmo potente. Un perceptrón multicapa consiste de mínimo 3 capas: capa de entrada capa oculta y capa de salida. Para este problema se varían las capas ocultas y las neuronas que tiene cada capa usando algunos valores entre 1 y 20 y entre 5 y 50 para cada parámetro respectivamente. Para los resultados completos ver apéndice 3. El mejor resultado obtenido fue:

#capas ocultas	Neuronas xcapa	Eficiencia	Sensibilidad	Precisión	F-Score	Error_Prueba	Tiempo de ejecución(seg)
2	**15**	0.9736	0.9736	0.9738	0.9735	0.0264	16.7362

Figura 6: Mejor resultado del algoritmo Perceptrón multicapa.

Figura 7:Matriz de confusión con 2 capas ocultas y 15 neuronas por capa

El perceptrón multicapa generó mejores resultados que el K vecinos más cercanos, casi un 1% menos de error, pero con un tiempo de ejecución mucho mayor, aun así es importante aclarar que para todos estos algoritmos los resultados pueden variar en diferentes ejecuciones debido a la aleatoriedad de la estrategia de validación bootstrapping Los parámetros usados fueron 2 capas ocultas y 15 neuronas por capa. En este modelo la clase C sigue siendo la que mayor cantidad de muestras fueron mal predichas con un total de 6 muestras.

- ***Random Forest:***

Random Forest, es una modificación del algoritmo de àrboles de clasificación, la cual consiste no sólo en crear un grupo de árboles *B*, sino también en incluir un componente aleatorio en la partición que se realiza en cada nodo. El componente aleatorio incluído en Random Forest es las escogencia del conjunto de variables a evaluar en cada nodo. La decisión final se toma a partir de de la combinación de decisiones de los *B* árboles. En nuestro caso, se eligen algunas cantidades de árboles entre 5 y 100, para el número de variables por nodo, varía entre algunos valores desde 2 hasta 19. Para los resultados completos ver apéndice 4. El mejor resultado obtenido fue:

#arboles	Variables xnodo	Eficiencia	Sensibilidad	Precisión	F-Score	Error_Prueba	Tiempo de ejecución(seg)
20	10	0.981	0.9801	0.9806	0.9802	0.019	0.3523

Figura 8: Mejor resultado del algoritmo Random Forest.

Figura 9: Matriz de confusión con 20 árboles y 10 variables analizadas por nodo

Esto modelo fue el que mejores resultados arrojó, con apenas 1.9% de error y con un tiempo de ejecución inferior al segundo, los parámetros utilizados fueron 20 árboles y 10 variables analizadas por nodo. Para la clase E, en la matriz de confusión, es la que falló más el modelo en las predicciones con 6 muestras mal clasificadas.

- ***Máquinas de Soporte Vectorial con kernel lineal y con kernel RBF:***

Para encontrar la mejor solución, este modelo utiliza como criterio la maximización del margen, el cual, se entiende como la distancia más corta entre la frontera de decisión y cualquiera de las muestras. Para este modelo, se crean vectores de soporte. Un vector de soporte es aquel que se encuentra en el punto más cercano a la frontera. SVM cuenta con 3 parámetros: Kernel, C y gamma. El kernel es la representación que ofrece una solución al problema de no poder usar una línea recta para la separación por medio de una proyección. Cuando la C es muy grande se le da mucho peso a los errores, provocando sobreajuste en el modelo y cuando la C es pequeña no se le da nada de peso a los errores entonces solo queda el margen, en este caso, para lograr un buen ajuste hay que aumentar mucho el tamaño del margen. El gamma es el coeficiente del kernel. En nuestro caso, el mejor resultado para el kernel lineal es C=300, gamma = 0 ya que gamma sólo se usa como coeficiente del kernel 'rbf', 'poly' y 'sigmoid. Para el kernel rbf es C=300 y gamma = 0.1. En las tablas el punto separa las cifras decimales, es decir 100.000 es 100 ó 1.000 es 1. Para los resultados completos ver apéndice 5.

Kernel	C	gamma	Eficiencia	Sensibilidad	Precisión	F-Score	Error_ Prueba	%VS	Tiempo ejecución (seg)
linear	**300.000**	**0.000**	0.9589	0.959	0.9593	0.9587	0.0411	0.157	0.5735
rbf	**300.000**	**0.100**	0.974	0.9736	0.9732	0.9733	0.026	0.1725	0.4044

Figura 10: Mejor resultado del algoritmo SVM con kernel lineal y RBF.

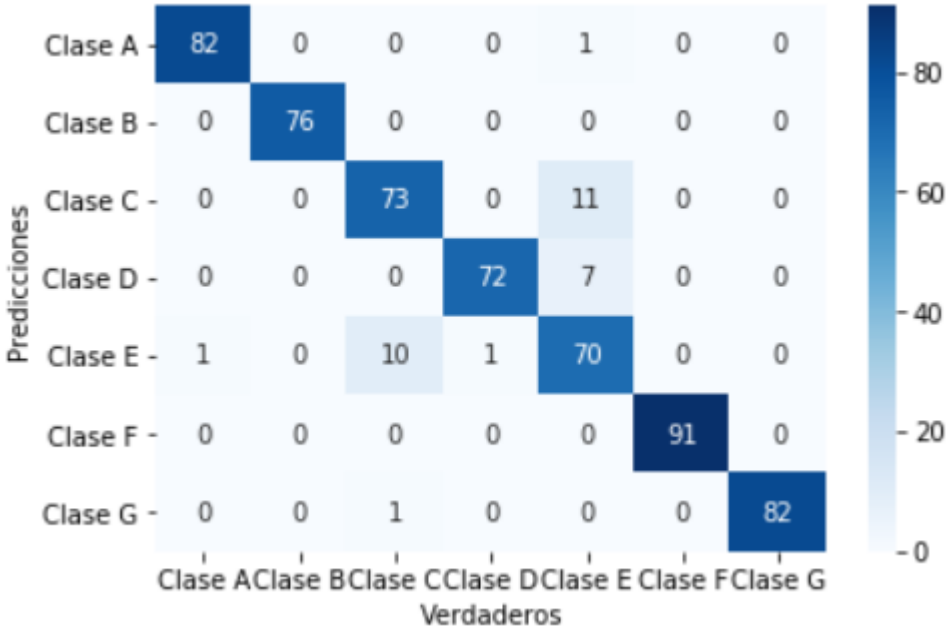

Figura 11: Matriz de confusión con kernel linear, C= 300 y gamma = 0

Figura 12: Matriz de confusión con kernel rbf, C = 300 y gamma = 0.1

Los mejores resultados fueron obtenidos con el kernel rbf con un error del 2.6% y un tiempo de ejecución de 0.40 segundos, incluso menor al modelo con kernel lineal. Para este problema el SVM también es bastante eficiente, también se observa que la clase C y la clase E siguen siendo las más problemáticas para acertar las predicciones.

C. Índice de Fisher:

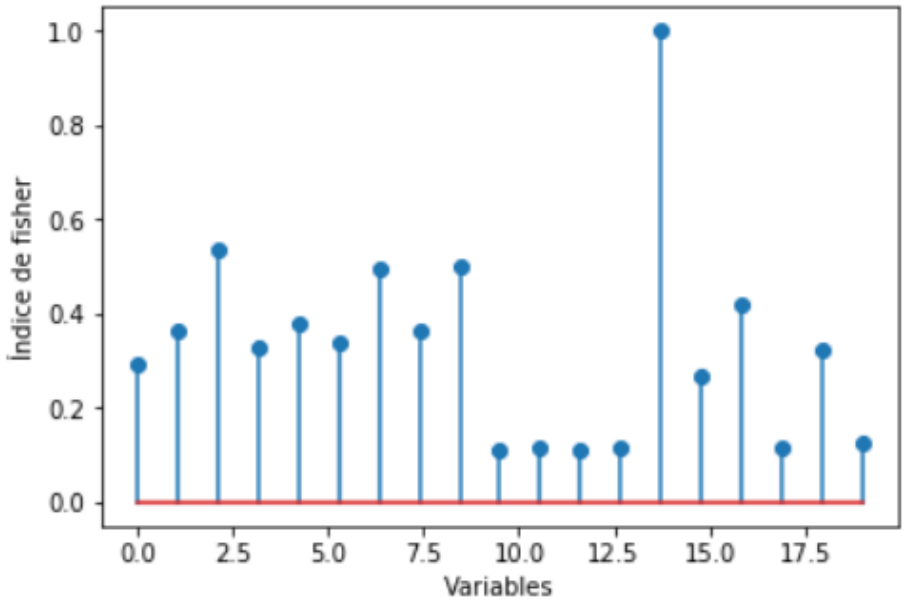

Figura 13: Resultado de índice de Fisher con cada característica..

Según los resultados del índice de Fisher, la variable con mayor discriminante es la variable 13 (exred-mean) y le siguen la 2 (region-pixel-count), la 6 (vegde-sd) y la 8 (hedge-sd). Por otro lado, las candidatas a ser eliminadas son las variables 9 (intensity-mean), 10 (rawred-mean), 11 (rawblue-mean) y 12 (rawgreen-mean) ya que son la que menor discriminación tienen, por debajo del 0.1. Por otro lado si se quiere poner el límite en mayor o igual a 0.5 solo quedarían las variables mencionadas al inicio, las demás son muy probables que se eliminen.

D. Selección de características:

Para la selección de características se usó la selección secuencial hacia atrás. Este método comienza con el conjunto completo de características y elimina una característica a la vez haciendo que el criterio J aumente o decresca mínimamente. Se usó una estrategia de búsqueda descendente y criterio de selección tipo wrapper con SVM con parámetros kernel rbf, C= 10, gamma= 1 y función de decisión OneVsOne.
Para los resultados completos ver apéndice 6.

#características	Eficiencia	Sensibilidad	Precisión	F-Score	Error_ Prueba	Tiempo de ejecución(seg)	Características
19	0.9697	0.9693	0.9697	0.9693	0.0303	0.2382	
14	0.9719	0.9703	0.9711	0.9705	0.0281	11.327	(0, 1, 3, 4, 5, 7, 9, 10, 12, 13, 14, 15, 17, 18)

Figura 14: Resultado de selección de características.

Con el total de las características se obtuvo un error del 3.03% mientras que con 14 características generó un error del 2.8% con un tiempo de ejecución de 11.3 segundos, bastante elevado para lo que se demora el modelo completo pero vale la pena pues se reduce la complejidad computacional en los modelos próximos a entrenar pero con el riesgo de que se haya perdido algo de información.
Comparando las características resultantes de este algoritmo con el análisis del cociente discriminante de Fisher, se mantiene la característica 13 que fue la de mayor índice de determinación, pero las otras 3 (2, 6 y 8) se eliminaron en esta etapa.

En las siguientes tablas se compara el resultado obtenido con las 14 características y con el dataset completo, se calcula la diferencia: errorInicial - errorConSelección y si da positivo hay una reducción en el error, si da negativo hay un aumento en el error, luego se calcula el porcentaje de reducción: 100-(errorConSelección*100/errorInicial), de igual manera si es positivo es una disminución en el error y si da negativo es que aumentó el error.

También se muestran algunos modelos que, con otros parámetros diferentes, consiguen mejores resultados, estos están resaltados de color naranja.

- ***K vecinos más cercanos:*** Para los resultados completos ver apéndice 7.

#vecinos	Eficiencia	Sensibilidad	Precisión	F-Score	Error_Prueba	Tiempo ejecución(seg)
1	0.9693	0.9699	0.9687	0.9689	0.0307	0.0821
Sin selección (1)	0.9658	0.9677	0.9661	0.9663	0.0342	0.0941
Reducción de error	Diferencia: 0.0035, Porcentaje: 10.23%		**Reducción del tiempo de ejecución**		Diferencia: 0.012, Porcentaje: 12.75%	

Figura 15: Mejor resultado de selección de características para K vecinos más cercanos.

- ***Perceptrón multicapa:*** Para los resultados completos ver apéndice 8.

#capas ocultas	Neuronasxcapa	Eficiencia	Sensibilidad	Precisión	F-Score	Error_Prueba	Tiempo de ejecución(seg)
2	**15**	0.965	0.9654	0.965	0.9648	0.035	17.3888
	50	0.9736	0.9726	0.9725	0.9724	0.0264	16.3969
Sin selección (2)	**15**	0.9736	0.9736	0.9738	0.9735	0.0264	16.7362
Reducción de error		Diferencia: -0.0086, Porcentaje: -32.57%		**Reducción del tiempo de ejecución**		Diferencia: -0.6526, Porcentaje: -3.9%	

Figura 16: Mejor resultado de selección de características para Perceptrón multicapa.

- ***Random Forest:*** Para los resultados completos ver apéndice 9.

#arboles	Variablesxnodo	Eficiencia	Sensibilidad	Precisión	F-Score	Error_Prueba	Tiempo ejecución(seg)
20	**10**	0.9632	0.977	0.9775	0.9771	0.0229	0.3793
100	**5**	0.9719	0.983	0.9835	0.983	0.0164	1.123
Sin selección(20)	**10**	0.981	0.9801	0.9806	0.9802	0.019	0.3523
Reducción de error		Diferencia: -0.004, Porcentaje: -20.53%		**Reducción del tiempo de ejecución**		Diferencia: -0.6526, Porcentaje: -3.9%	

Figura 17: Mejor resultado de selección de características para Random Forest.

- ***Máquinas de Soporte Vectorial con kernel lineal y con kernel RBF:*** Para los resultados completos ver apéndice 10.

Kernel	C	gamma	Eficiencia	Sensibilidad	Precisión	F-Score	Error_Prueba	%VS	Tiempo ejecución (seg)
Linear	**300**	**0.000**	0.955	0.9555	0.9567	0.9559	0.045	0.1516	0.4644
	100	**1.000**	0.9736	0.9744	0.9744	0.9741	0.0264	0.1572	0.3583
rbf	**300**	**0.100**	0.9693	0.9696	0.9705	0.9698	0.0307	0.1682	0.3603
Sin selección	**300**	**0.000**	0.9589	0.959	0.9593	0.9587	0.0411	0.157	0.5735

(Linear)									
Sin selección (rbf)	**300**	**0.100**	0.974	0.9736	0.9732	0.9733	0.026	0.1725	0.4044
Reducción del error con kernel rbf			Diferencia: -0.0047, Porcentaje: -18.08%			**Reducción del tiempo con kernel rbf**		Diferencia: 0.044 seg, Porcentaje: 10.9%	
Reducción del error con kernel lineal			Diferencia: -0.0039, Porcentaje: -9.5%			**Reducción del tiempo de ejecución con kernel lineal**		Diferencia: 0.109 seg, Porcentaje: 19.8%	

Figura 18: Mejor resultado de selección de características para Máquinas de soporte vectorial.

E. Extracción de características PCA:

PCA es una técnica de extracción cuyo objetivo es reducir la dimensión de un conjunto de variables, conservando la mayor cantidad de información que sea posible, esta información está asociada al nivel de variación de las variables. Para poder realizar la extracción de características, la media de los datos debe ser 0. Una vez que los datos están escalados y tengan media 0, se obtiene la matriz de covarianza; después, se calculan los valores propios de la matriz y sus respectivos vectores propios. Para finalizar se ordenan los valores propios de manera descendente y se proyectan los datos sobre las direcciones principales.

Al igual que en el punto anterior, se calcula la diferencia de los errores comparando el error generado por el modelo usando todo el dataset y el error generado por el modelo usando nComponentes. En algunos modelos se subrayan varios resultados, esto porque ambos son interesantes para tener en cuenta y se tomará la decisión de cuál es el mejor dependiendo del error en la prueba pues esta medida representa que tan bien o que tan mal predijo el modelo y dependiendo, también, del total de componentes, pues entre menor número de estas será mejor.

- **K vecinos más cercanos**: Para los resultados completos ver apéndice 11.

Modelo usado: KNeighborsClassifier(n_neighbors = 1)

#Componentes	Eficiencia	Sensibilidad	Precisión	F-Score	Error_Prueba	Tiempo ejecución(seg)
1	0.5584	0.5636	0.5594	0.5595	0.4416	0.6906
2	0.6466	0.6493	0.6493	0.6478	0.3534	0.7387
3	0.8421	0.8429	0.8419	0.8417	0.1579	0.7146
4	0.8932	0.8914	0.892	0.8907	0.1068	0.7617
5	0.9161	0.9171	0.9169	0.9159	0.0839	0.7136
6	0.9252	0.9233	0.9239	0.9229	0.0748	0.7106
7	0.9386	0.9387	0.9388	0.938	0.0614	0.7177
8	0.9386	0.9387	0.939	0.9384	0.0614	0.7397
9	0.9516	0.9531	0.9516	0.9518	0.0484	0.7066
10	0.9477	0.9493	0.9497	0.949	0.0523	0.7137
11	0.9593	0.9587	0.9595	0.9587	0.0407	0.7197
12	0.9572	0.9564	0.9573	0.9564	0.0428	0.7106
13	0.9585	0.9592	0.9588	0.9583	0.0415	0.7357
14	0.9667	0.9678	0.9672	0.967	0.0333	0.7767
15	0.9554	0.9572	0.9551	0.9555	0.0446	0.7247

16	0.9606	0.9596	0.9599	0.9595	0.0394	0.7507
17	0.9702	0.9687	0.9697	0.9689	0.0298	0.7527
18	0.955	0.9549	0.9565	0.9547	0.045	0.7487
19	0.9658	0.965	0.9649	0.9645	0.0342	0.7717
Reducción de error con 14 componentes:	Diferencia: 0.0009, Porcentaje: 2.63%		Reducción de tiempo con 14 componentes		Diferencia: -0.005, Porcentaje: -0.648%	
Reducción de error con 17 componentes:	Diferencia: 0.0044, Porcentaje: 12.866%		Reducción de tiempo con 14 componentes		Diferencia: -0.019, Porcentaje: 2.462%	

Figura 19: Mejor resultado de extracción de características para K vecinos más cercanos.

Las dos posibles opciones para K vecinos más cercanos son con 14 y 17 componentes, ambos presenta una disminución del error pero es mayor con 14 componentes, aunque es muy pequeña la diferencia y en una diferente ejecución del modelo estos valores pueden cambiar, aún así, se mueven en intervalos pequeños en donde el resultado no cambia drásticamente.

- **Perceptrón multicapa:** Para los resultados completos ver apéndice 12.

Modelo usado:
MLPClassifier(activation='tanh',max_iter = 1000,hidden_layer_sizes=(15,15))

#Componentes	Eficiencia	Sensibilidad	Precisión	F-Score	Error_Prueba	Tiempo ejecución(seg)
1	0.5484	0.5428	0.5196	0.5142	0.4516	9.6608
2	0.673	0.6745	0.6792	0.6688	0.327	20.4336
3	0.8287	0.8292	0.8309	0.8271	0.1713	19.9472
4	0.8858	0.8879	0.8887	0.8863	0.1142	21.1142
5	0.9057	0.9035	0.9052	0.9029	0.0943	21.0822
6	0.9178	0.9161	0.917	0.915	0.0822	21.1212
7	0.9446	0.9448	0.9443	0.9441	0.0554	20.7058
8	0.9403	0.9386	0.9386	0.9381	0.0597	21.2293
9	0.9572	0.9566	0.957	0.9565	0.0428	20.1894
10	0.9585	0.9579	0.9577	0.9575	0.0415	22.1802
11	0.965	0.965	0.9657	0.9652	0.035	19.7209
12	0.9524	0.9532	0.9529	0.9526	0.0476	20.2484
13	0.9693	0.9689	0.9689	0.9686	0.0307	18.6089
14	0.9684	0.9689	0.9691	0.9689	0.0316	18.666
15	0.9715	0.9712	0.9709	0.9709	0.0285	18.5959
16	0.9723	0.9714	0.9734	0.9722	0.0277	18.4868
17	0.9697	0.9699	0.9701	0.9698	0.0303	16.6351
18	0.9788	0.9794	0.9785	0.9788	0.0212	18.7441

19	0.9749	0.9753	0.9759	0.9755	0.0251	19.5058
Reducción de error con 16 componentes:	Diferencia: -0.0026, Porcentaje: -10.36%		Reducción de tiempo con 16 componentes		Diferencia: 1.019, Porcentaje: 5.22%	
Reducción de error con 18 componentes:	Diferencia: 0.0039, Porcentaje: 15.54%		Reducción de tiempo con 18 componentes		Diferencia: 0.762, Porcentaje: 3.9%	

Figura 20: Mejor resultado de extracción de características para Perceptrón multicapa.

En este modelo el mejor resultado fue generado teniendo 18 componentes, como es practicamente igual al tamaño original, se considera también el resultado obtenido con 16 componentes, pues la diferencia de error no es muy grande, el tiempo de ejecución es casi igual y al ser menor cantidad de dimensiones, la complejidad del problema disminuye.

- **Random Forest:** Para los resultados completos ver apéndice 13.

Modelo usado: RandomForestClassifier(n_estimators = 20, max_features = 10)

#Componentes	Eficiencia	Sensibilidad	Precisión	F-Score	Error_Prueba	Tiempo de ejecución(seg)
10	0.9394	0.9394	0.9389	0.9389	0.0606	1.3122
11	0.942	0.9398	0.9407	0.94	0.058	1.3092
12	0.936	0.937	0.9367	0.9362	0.064	1.3012
13	0.9412	0.9398	0.9404	0.9398	0.0588	1.2792
14	0.9459	0.9468	0.9473	0.9467	0.0541	1.2611
15	0.9507	0.9502	0.9508	0.95	0.0493	1.3122
16	0.9438	0.9451	0.9448	0.9444	0.0562	1.3252
17	0.9537	0.9549	0.9544	0.9541	0.0463	1.2882
18	0.952	0.9519	0.9525	0.952	0.048	1.1991
19	0.9715	0.9714	0.9718	0.9714	0.0285	1.2922
Reducción de error con 15 componentes:	Diferencia: -0.0178, Porcentaje: -62.46%		Reducción de tiempo con 16 componentes		Diferencia: -0.004, Porcentaje: 0.31%	

Figura 21: Mejor resultado de extracción de características para Random Forest.

Para random forest el mejor resultado con respecto al error fue con 17 componentes, sin embargo, este resultado es mayor que el error obtenido usando todo el conjunto de datos, por lo que para este modelo no es apropiado reducir la dimensionalidad del problema.

- ***Máquinas de Soporte Vectorial con kernel lineal y con kernel RBF***: Para los resultados completos ver apéndice 14.

Modelo usado: SVC(gamma=0.1, C=float(300), kernel='rbf', decision_function_shape='ovo' , probability = True)

#Componentes	Eficiencia	Sensibilidad	Precisión	F-Score	Error_Prueba	%VS	Tiempo ejecución(seg)
1	0.5294	0.5472	0.5647	0.4993	0.4706	0.7536	2.3392

2	0.6315	0.6436	0.6678	0.6299	0.3685	0.6686	1.9898
3	0.8452	0.8452	0.8484	0.8415	0.1548	0.3801	1.5794
4	0.8659	0.8681	0.8762	0.8654	0.1341	0.3145	1.5884
5	0.9066	0.9041	0.9095	0.9032	0.0934	0.303	1.6755
6	0.92	0.9187	0.9207	0.9184	0.08	0.2679	1.5074
7	0.9446	0.9434	0.9436	0.9431	0.0554	0.2561	1.5103
8	0.9459	0.945	0.945	0.9444	0.0541	0.2386	1.3826
9	0.9516	0.9527	0.9523	0.9519	0.0484	0.2309	1.3462
10	0.9563	0.9556	0.9558	0.9553	0.0437	0.2423	1.3622
11	0.9654	0.964	0.9653	0.9642	0.0346	0.2315	1.3622
12	0.9615	0.9615	0.961	0.9609	0.0385	0.2285	1.3242
13	0.9628	0.9638	0.9632	0.963	0.0372	0.2164	1.342
14	0.9676	0.9684	0.9678	0.9679	0.0324	0.2156	1.3128
15	0.9537	0.9542	0.9547	0.9537	0.0463	0.2141	1.3311
16	0.9667	0.9676	0.9674	0.9673	0.0333	0.2201	1.3181
17	0.9676	0.9677	0.967	0.967	0.0324	0.2165	1.3428
18	0.9663	0.9655	0.9658	0.9653	0.0337	0.2159	1.3312
19	0.9702	0.9708	0.9707	0.9704	0.0298	0.2168	1.4273
Reducción de error con 14 componentes:		Diferencia: -0.0298, Porcentaje: -9.724%		Reducción de tiempo con 14 componentes		Diferencia: 0.115, Porcentaje: 8.022%	

Figura 22: Mejor resultado de extracción de características para Máquina de soporte vectorial lineal y RBF.

Sucede lo mismo que en el caso anterior con SVM, el error en las pruebas con menos componentes fue mayor, pero en este caso la diferencia no es mucha con los resultados con 14 y 17 componentes, sin embargo, las métricas obtenidas con 14 componentes son un poco mejores que las obtenidas con 17, por lo tanto es conveniente usar 14 componentes en este modelo.

5. Análisis

Algunos de los autores que trabajaron con esta base de datos, tomaron la partición propuesta por los datos de train con 210 muestras y test con 2100.

En el artículo Vlassis, N., & Likas, A. se realizó un análisis de caraterísticas principales como método de extracción mientras que en este trabajo se utilizó el análisis de características principales como método de selección. Por otro lado, en el artículo Tung, A. K. H., Xu, X., & Ooi, B. C. 2005, se eliminan las características 5, 7 y 9 por la redundancia con las características 4, 6 y 8; a pesar de esto, aplicando una selección de características hacia atrás, resultó como mejor opción eliminar las características 5, 7 y 9. Por otro lado, en el artículo de J.T.-Y. Kwok, 1999, con la implementación de SVM con salidas moderadas se obtuvo un error del 8.4%, usando SVM con un *evidence framework* se obtuvo un error del 9.8% y en este trabajo, los mejores resultados con kernel lineal y rbf se obtuvo un error del 4.1% y 2.6% respectivamente. Lindenbaum, M. Markovitch, S & Rusakov, D. 1999 implementa KNN con un *random field model* que sirve para encontrar la muestra con mayor utilidad considerando su efecto en el clasificador, se obtuvo un error del 10.9%, en este trabajo se utilizó el KNN general obteniendo un error del 3.4%. Por último, en el artículo Albert Ribes, 2019 se utiliza esta base de datos en los algoritmo SVM y *Random Forest* con un error de 4.2% y 2.8% respectivamente, en cambio, en este trabajo se obtuvo 4.1% y 1.9% en el mismo orden.

El mejor resultado que se obtuvo en este trabajo fue Random forest con 20 árboles y 10 variables por nodo

Referencias

[1] N. Vlassis y A. Likas, “A Greedy EM Algorithm for Gaussian Mixture Learning”, 2002.

[2] J. T.-Y. Kwok, “Moderating the outputs of support vector machine classifiers”, en IJCNN’99. International Joint Conference on Neural Networks. Proceedings (Cat. No.99CH36339), 1999, vol. 2, pp. 943–948.

[3] M. Lindenbaum, S. Markovitch, y D. Rusakov, “Selective Sampling Using Random Field Modelling”, 1999.

[4] A. K. H. Tung, X. Xu, y B. C. Ooi, “CURLER: Finding and Visualizing Nonlinear Correlation Clusters”, en Proceedings of the 2005 ACM SIGMOD international conference on Management of data - SIGMOD ’05, 2005, p. 467.

[5] A. Ribes, “Using Random Fourier Features with Random Forest”, 2019.

Apéndices:

- ***Apéndice 1: Naïve Bayes***

Eficiencia	Int_ Eficiencia	Sensibilidad	Int_ Sensibilidad	Precisión	Int_ Precision	F-Score	Int_ F-Score	Error_ Prueba	Int_Error	Tiempo de ejecución(seg)
0.7967	0.0176	0.7972	0.2745	0.8153	0.2006	0.7789	0.2522	0.2033	0.0176	0.026

- ***Apéndice 2: K vecinos más cercanos***

Número vecinos	Eficiencia	Int_ Eficiencia	Sensibilidad	Int_ Sensibilidad	Precision	Int_ Precision	F-Score	Int_ F-Score	Error_ Prueba	Int_ Error	Tiempo ejecución (seg)
1	0.9658	0.0062	0.9677	0.0463	0.9661	0.044	0.9663	0.0388	0.0342	0.0062	0.0941
2	0.9537	0.0111	0.9531	0.071	0.9541	0.0496	0.9525	0.0551	0.0463	0.0111	0.0991
3	0.9572	0.0039	0.9566	0.0488	0.957	0.0458	0.9564	0.0434	0.0428	0.0039	0.1011
4	0.9494	0.0033	0.9503	0.068	0.95	0.0484	0.9491	0.0513	0.0506	0.0033	0.1001
5	0.9546	0.0059	0.9547	0.053	0.9546	0.045	0.9543	0.0458	0.0454	0.0059	0.1111
6	0.9503	0.0051	0.9506	0.0614	0.9519	0.0415	0.9504	0.0449	0.0497	0.0051	0.1061
7	0.952	0.0054	0.9522	0.0555	0.9522	0.0471	0.9515	0.0449	0.048	0.0054	0.1101
8	0.9399	0.0078	0.9398	0.0686	0.9413	0.0577	0.9398	0.0585	0.0601	0.0078	0.1131
9	0.9429	0.01	0.9441	0.063	0.9443	0.0538	0.9431	0.0509	0.0571	0.01	0.1121
10	0.9446	0.0124	0.945	0.0712	0.945	0.057	0.9438	0.0561	0.0554	0.0124	0.1111
11	0.9412	0.0042	0.9422	0.0632	0.9413	0.057	0.941	0.055	0.0588	0.0042	0.1131
12	0.9403	0.0045	0.9409	0.0689	0.9405	0.0558	0.9399	0.0573	0.0597	0.0045	0.1151
13	0.9477	0.0123	0.9462	0.0644	0.9467	0.0532	0.9456	0.0523	0.0523	0.0123	0.1221
14	0.9347	0.011	0.937	0.081	0.9349	0.0661	0.9343	0.0653	0.0653	0.011	0.1151
15	0.9433	0.0046	0.9427	0.0588	0.9434	0.0531	0.9425	0.0518	0.0567	0.0046	0.1281
16	0.9321	0.0055	0.934	0.071	0.9334	0.0557	0.9329	0.0585	0.0679	0.0055	0.1221

17	0.9343	0.0116	0.9352	0.0771	0.935	0.0558	0.9342	0.0612	0.0657	0.0116	0.1301
18	0.9433	0.0058	0.9406	0.0691	0.9415	0.0594	0.9402	0.0587	0.0567	0.0058	0.1291
19	0.9282	0.0066	0.9281	0.0833	0.9276	0.0629	0.9268	0.0677	0.0718	0.0066	0.1301
20	0.9343	0.003	0.9329	0.075	0.9338	0.0617	0.9323	0.0617	0.0657	0.003	0.1311

- ***Apéndice 3: Redes Neuronales Artificiales***

#capas ocultas	Neuronas xcapa	Eficiencia	Int_Eficiencia	Sensibilidad	Int_Sensibilidad	Precision	Int_ Precision	F-Score	Int_ F-Score	Error_ Prueba	Int_ Error	Tiempo ejecución (seg)
1	**5**	0.9403	0.0075	0.9414	0.0681	0.9427	0.0763	0.9411	0.0676	0.0597	0.0075	12.2802
	10	0.952	0.0054	0.952	0.0563	0.9527	0.0548	0.9521	0.0536	0.048	0.0054	13.192
	15	0.9567	0.0071	0.9579	0.0472	0.9582	0.0533	0.9579	0.0488	0.0433	0.0071	14.4902
	20	0.955	0.0035	0.9538	0.0533	0.955	0.0546	0.9539	0.0497	0.045	0.0035	15.5431
	25	0.9654	0.0087	0.9663	0.0401	0.9663	0.0468	0.9659	0.0391	0.0346	0.0087	15.8805
	30	0.9516	0.0081	0.9542	0.058	0.9535	0.0554	0.9534	0.0534	0.0484	0.0081	17.0425
	35	0.9676	0.005	0.9665	0.0377	0.9673	0.0373	0.9666	0.0339	0.0324	0.005	17.3708
	40	0.9559	0.0133	0.9555	0.0577	0.9544	0.0535	0.9545	0.0526	0.0441	0.0133	17.8713
	45	0.9632	0.0067	0.9636	0.0434	0.9647	0.045	0.9638	0.0411	0.0368	0.0067	17.2797
	50	0.9576	0.0045	0.9579	0.0452	0.9588	0.0588	0.9579	0.0489	0.0424	0.0045	18.0624
2	**5**	0.949	0.011	0.9478	0.0593	0.9488	0.0641	0.9479	0.0598	0.051	0.011	16.0566
	10	0.9598	0.0048	0.9597	0.0419	0.9597	0.0501	0.9593	0.0431	0.0402	0.0048	15.7774
	15	0.9736	0.0058	0.9736	0.0305	0.9738	0.0332	0.9735	0.029	0.0264	0.0058	16.7362
	20	0.9736	0.0058	0.9736	0.0318	0.9735	0.0315	0.9735	0.0304	0.0264	0.0058	17.3858
	25	0.9723	0.0032	0.9722	0.029	0.972	0.0328	0.972	0.0286	0.0277	0.0032	16.9444
	30	0.9615	0.0072	0.9621	0.0415	0.9624	0.0493	0.9621	0.0436	0.0385	0.0072	16.445
	35	0.9654	0.0017	0.9658	0.0422	0.9659	0.0411	0.9657	0.0403	0.0346	0.0017	19.3326
	40	0.9667	0.007	0.9666	0.0402	0.9672	0.0407	0.9667	0.0384	0.0333	0.007	20.1824
	45	0.9697	0.0051	0.9702	0.0408	0.9699	0.0419	0.9696	0.0366	0.0303	0.0051	15.9295
	50	0.9689	0.0068	0.9681	0.0366	0.9682	0.042	0.9679	0.0367	0.0311	0.0068	14.5663

- ***Apéndice 4: Random Forest***

#arboles	Variables xnodo	Eficiencia	Int_Eficiencia	Sensibilidad	Int_Sensibilidad	Precision	Int_Precision	F-Score	Int_ F-Score	Error_ Prueba	Int_ Error	Tiempo ejecución (seg)
5	**2**	0.9567	0.005	0.9557	0.0401	0.9551	0.0478	0.9549	0.0393	0.0433	0.005	0.05
	5	0.9667	0.0063	0.9666	0.0358	0.9669	0.0387	0.9664	0.0331	0.0333	0.0063	0.0661
	10	0.9663	0.0045	0.9659	0.0439	0.9663	0.0368	0.9656	0.0353	0.0337	0.0045	0.0991

	13	0.9615	0.0048	0.9628	0.0453	0.9624	0.0428	0.9622	0.0405	0.0385	0.0048	0.1171
	15	0.9654	0.0017	0.9666	0.0347	0.9669	0.0345	0.9665	0.0321	0.0346	0.0017	0.1331
	19	0.9589	0.015	0.9598	0.0446	0.9606	0.0559	0.9596	0.0459	0.0411	0.015	0.1621
10	**2**	0.955	0.0024	0.9557	0.0451	0.9569	0.0468	0.9559	0.0426	0.045	0.0024	0.0831
	5	0.9753	0.0022	0.9755	0.0317	0.9755	0.0295	0.9754	0.0294	0.0247	0.0022	0.1241
	10	0.971	0.0022	0.9713	0.0358	0.971	0.0302	0.971	0.0301	0.029	0.0022	0.1862
	13	0.9732	0.0031	0.9728	0.0278	0.973	0.0315	0.9727	0.0271	0.0268	0.0031	0.2212
	15	0.9658	0.0081	0.966	0.0367	0.9663	0.0388	0.9659	0.0353	0.0342	0.0081	0.2522
	19	0.9697	0.007	0.969	0.0402	0.9689	0.0431	0.9687	0.0386	0.0303	0.007	0.3093
20	**2**	0.9689	0.0037	0.9695	0.0404	0.9693	0.037	0.9692	0.0365	0.0311	0.0037	0.1501
	5	0.9762	0.0093	0.9762	0.0293	0.9766	0.028	0.9763	0.027	0.0238	0.0093	0.2232
	10	0.981	0.0065	0.9801	0.0258	0.9806	0.0258	0.9802	0.0233	0.019	0.0065	0.3523
	13	0.9775	0.0037	0.977	0.0285	0.9769	0.0274	0.9767	0.0249	0.0225	0.0037	0.4304
	15	0.9723	0.0042	0.9728	0.0257	0.9726	0.0339	0.9725	0.0267	0.0277	0.0042	0.4834
	19	0.9728	0.0069	0.9713	0.0333	0.9724	0.0273	0.9716	0.0276	0.0272	0.0069	0.5985
50	**2**	0.971	0.0064	0.9728	0.0314	0.9723	0.0332	0.9724	0.0291	0.029	0.0064	0.3543
	5	0.981	0.0047	0.9813	0.025	0.9805	0.0241	0.9807	0.0206	0.019	0.0047	0.5415
	10	0.9775	0.005	0.9773	0.0273	0.9771	0.0256	0.977	0.0233	0.0225	0.005	0.8678
	13	0.9792	0.0039	0.9801	0.0238	0.9799	0.0243	0.9799	0.0226	0.0208	0.0039	1.056
	15	0.9758	0.0074	0.9759	0.0321	0.976	0.0284	0.9757	0.0261	0.0242	0.0074	1.1971
	19	0.9706	0.0078	0.9706	0.0382	0.9722	0.0393	0.9709	0.0332	0.0294	0.0078	1.4823
100	**2**	0.974	0.0065	0.974	0.033	0.9748	0.0373	0.9741	0.031	0.026	0.0065	0.6866
	5	0.9779	0.0078	0.9772	0.0278	0.9774	0.0291	0.9772	0.0267	0.0221	0.0078	1.066
	10	0.9771	0.0086	0.9772	0.0287	0.9774	0.0301	0.9771	0.0258	0.0229	0.0086	1.7076
	13	0.9766	0.0026	0.9762	0.0346	0.9759	0.0263	0.9759	0.0279	0.0234	0.0026	2.1129
	15	0.981	0.0053	0.9803	0.0248	0.981	0.0248	0.9804	0.0206	0.019	0.0053	2.3501
	19	0.9693	0.0054	0.9703	0.0297	0.971	0.032	0.9704	0.0282	0.0307	0.0054	2.9317

- ***Apéndice 5: Máquinas de Soporte Vectorial con kernel lineal y con kernel RBF.***

Kernel	C	gamma	Eficiencia	Int_Efi	Sens	Int_Sen	Prec	Int_Prec	F-Score	Int_F-Score	Error_Prueba	Int_Error	%VS	Tiempo ejecución (seg)
linear	**0.001**	**0.000**	0.2228	0.0524	0.2397	0.417	0.0847	0.2017	0.1097	0.2135	0.7772	0.0524	0.9993	1.5474
	0.010	**0.000**	0.5562	0.0344	0.5779	0.4497	0.5502	0.3945	0.4813	0.385	0.4438	0.0344	0.945	1.2501
	0.100	**0.000**	0.8737	0.0153	0.875	0.1214	0.8754	0.1169	0.8743	0.1164	0.1263	0.0153	0.6876	0.5815
	1.000	**0.000**	0.9208	0.0074	0.9234	0.0957	0.9242	0.0951	0.9234	0.094	0.0792	0.0074	0.3952	0.3373

	10.000	**0.000**	0.939	0.0041	0.94	0.0762	0.9389	0.0683	0.9391	0.0705	0.061	0.0041	0.233	0.3113
	100.000	**0.000**	0.9485	0.0103	0.9495	0.055	0.9509	0.0688	0.9495	0.0583	0.0515	0.0103	0.1713	0.3803
	200.000	**0.000**	0.949	0.007	0.9489	0.0547	0.9502	0.0755	0.9489	0.062	0.051	0.007	0.163	0.4914
	300.000	**0.000**	0.9589	0.0065	0.959	0.048	0.9593	0.0531	0.9587	0.0467	0.0411	0.0065	0.157	0.5735
rbf	**0.001**	**0.001**	0.1207	0.0026	0.1429	0.3499	0.0172	0.0422	0.0308	0.0754	0.8793	0.0026	1	2.8066
		0.010	0.1566	0.0575	0.1776	0.381	0.0325	0.0778	0.0535	0.1239	0.8434	0.0575	1	2.8216
		0.100	0.1241	0.0051	0.1429	0.3499	0.0177	0.0435	0.0315	0.0773	0.8759	0.0051	1	2.8096
		1.000	0.1488	0.047	0.1722	0.3706	0.0357	0.099	0.0554	0.1404	0.8512	0.047	0.9993	2.8086
	0.010	**0.001**	0.1873	0.1119	0.2054	0.3943	0.0591	0.1417	0.0838	0.1827	0.8127	0.1119	0.999	2.7905
		0.010	0.1254	0.0053	0.1429	0.3499	0.0179	0.0439	0.0318	0.0781	0.8746	0.0053	1	2.7985
		0.100	0.1267	0.0097	0.1442	0.3494	0.0894	0.2563	0.0345	0.0778	0.8733	0.0097	1	2.8005
		1.000	0.5536	0.035	0.5794	0.4531	0.5274	0.371	0.4801	0.3881	0.4464	0.035	0.9597	2.5693
	0.100	**0.001**	0.1246	0.01	0.1429	0.3499	0.0178	0.0438	0.0316	0.0777	0.8754	0.01	1	2.8156
		0.010	0.2435	0.014	0.2659	0.4131	0.1712	0.3046	0.1617	0.276	0.7565	0.014	0.9994	2.8226
		0.100	0.7716	0.0231	0.7807	0.2352	0.8082	0.188	0.7656	0.1914	0.2284	0.0231	0.8829	1.8577
		1.000	0.8953	0.0082	0.8942	0.1165	0.8929	0.106	0.8927	0.1089	0.1047	0.0082	0.6467	1.0339
	1.000	**0.001**	0.2249	0.028	0.2455	0.4059	0.1621	0.3452	0.15	0.2896	0.7751	0.028	1	2.7915
		0.010	0.7833	0.0304	0.7889	0.2235	0.8147	0.1764	0.7748	0.1775	0.2167	0.0304	0.8806	1.7766
		0.100	0.8906	0.0136	0.8913	0.1215	0.8921	0.117	0.8903	0.116	0.1094	0.0136	0.5878	0.8077
		1.000	0.9334	0.0106	0.9355	0.098	0.9353	0.083	0.933	0.0829	0.0666	0.0106	0.3313	0.4904
	10.000	**0.001**	0.7574	0.0314	0.7661	0.249	0.7922	0.1711	0.7461	0.1878	0.2426	0.0314	0.8786	1.7716
		0.010	0.8901	0.0144	0.8911	0.1252	0.8901	0.1197	0.8898	0.1203	0.1099	0.0144	0.5863	0.7977
		0.100	0.9308	0.0032	0.9324	0.0874	0.9322	0.087	0.9314	0.0841	0.0692	0.0032	0.3242	0.4454
		1.000	0.965	0.0114	0.9647	0.0449	0.9647	0.0456	0.9643	0.0418	0.035	0.0114	0.2142	0.4044
	100.000	**0.001**	0.8849	0.0119	0.8839	0.126	0.8823	0.1227	0.8825	0.1224	0.1151	0.0119	0.5921	0.7967
		0.010	0.9321	0.0033	0.9343	0.0882	0.934	0.0816	0.9329	0.0799	0.0679	0.0033	0.3272	0.4634
		0.100	0.9619	0.012	0.9629	0.052	0.9627	0.0446	0.9624	0.0459	0.0381	0.012	0.1985	0.3923
		1.000	0.965	0.0043	0.9659	0.0448	0.965	0.0486	0.9648	0.0406	0.035	0.0043	0.1709	0.4074
	200.000	**0.001**	0.9131	0.0155	0.9157	0.1124	0.9164	0.1024	0.9147	0.1042	0.0869	0.0155	0.4961	0.6496
		0.010	0.9407	0.0069	0.9385	0.0758	0.9389	0.0831	0.9381	0.0768	0.0593	0.0069	0.2846	0.4454
		0.100	0.965	0.0055	0.9652	0.0396	0.9645	0.0478	0.9645	0.041	0.035	0.0055	0.1806	0.3874
		1.000	0.9706	0.0053	0.9708	0.0413	0.9713	0.0409	0.9706	0.0364	0.0294	0.0053	0.1666	0.4334
	300.000	**0.001**	0.9208	0.0062	0.9227	0.0997	0.9226	0.0943	0.9216	0.0935	0.0792	0.0062	0.4544	0.5963
		0.010	0.9446	0.0075	0.944	0.0843	0.9428	0.0724	0.9423	0.0738	0.0554	0.0075	0.2598	0.4317
		0.100	0.974	0.0021	0.9736	0.0326	0.9732	0.0341	0.9733	0.0321	0.026	0.0021	0.1725	0.4044

		1.000	0.9728	0.005	0.9728	0.0363	0.9723	0.035	0.9724	0.0333	0.0272	0.005	0.1634	0.4504

Apéndice 6: Selección de características

#características	Eficiencia	Int_Eficiencia	Sensibilidad	Int_Sensibilidad	Precision	Int_Precision	F-Score	Int_F-Score	Error_Prueba	Int_error	Tiempo ejecución (seg)	Características
19	0.9697	0.0029	0.9693	0.0338	0.9697	0.0393	0.9693	0.0345	0.0303	0.0029	0.2382	
18	0.9593	0.009	0.9599	0.0448	0.9604	0.0497	0.9598	0.0444	0.0407	0.009	2.7742	(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 11, 12, 13, 14, 15, 16, 17, 18)
17	0.9619	0.008	0.9639	0.0421	0.9651	0.0489	0.9643	0.0435	0.0381	0.008	5.3096	(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 15, 17, 18)
16	0.9676	0.0045	0.9678	0.0411	0.968	0.0432	0.9675	0.0379	0.0324	0.0045	7.4259	(0, 1, 3, 5, 6, 7, 8, 9, 10, 11, 13, 14, 15, 16, 17, 18)
15	0.9667	0.0081	0.967	0.0377	0.9673	0.0461	0.967	0.0405	0.0333	0.0081	9.3125	(0, 1, 3, 4, 5, 8, 9, 10, 11, 13, 14, 15, 16, 17, 18)
14	0.9719	0.0039	0.9703	0.0359	0.9711	0.0387	0.9705	0.0347	0.0281	0.0039	11.327	(0, 1, 3, 4, 5, 7, 9, 10, 12, 13, 14, 15, 17, 18)
13	0.9645	0.0061	0.9644	0.0529	0.964	0.0466	0.9635	0.044	0.0355	0.0061	13.1923	(0, 1, 3, 4, 7, 9, 10, 11, 13, 15, 16, 17, 18)
12	0.9654	0.0074	0.9659	0.0437	0.9659	0.048	0.9656	0.0428	0.0346	0.0074	14.5797	(0, 1, 2, 4, 5, 7, 8, 10, 13, 15, 17, 18)
11	0.9641	0.0037	0.9632	0.047	0.9633	0.0451	0.9631	0.0448	0.0359	0.0037	16.4635	(0, 1, 3, 5, 7, 9, 11, 13, 15, 17, 18)
10	0.9606	0.0039	0.9592	0.0493	0.9605	0.0531	0.9595	0.0482	0.0394	0.0039	18.3724	(0, 1, 3, 5, 7, 11, 13, 15, 17, 18)
9	0.9654	0.0024	0.966	0.0418	0.9671	0.045	0.9663	0.0407	0.0346	0.0024	19.2867	(0, 1, 6, 9, 13, 14, 15, 17, 18)
8	0.9615	0.01	0.9613	0.0429	0.9628	0.0526	0.9618	0.0456	0.0385	0.01	20.3162	(0, 1, 3, 9, 13, 15, 17, 18)
7	0.9641	0.0067	0.9622	0.0451	0.9631	0.0492	0.9625	0.0456	0.0359	0.0067	21.0221	(0, 1, 5, 13, 15, 17, 18)
6	0.9676	0.0039	0.967	0.0394	0.967	0.0471	0.9669	0.0422	0.0324	0.0039	23.1738	(0, 1, 7, 13, 15, 17)
5	0.9667	0.0054	0.9663	0.0439	0.9667	0.0507	0.9659	0.0413	0.0333	0.0054	23.42	(0, 1, 13, 15, 17)
4	0.9516	0.011	0.9517	0.0602	0.9529	0.0741	0.9516	0.0641	0.0484	0.011	23.6162	(1, 13, 15, 17)
3	0.9204	0.0097	0.9186	0.0772	0.9214	0.0729	0.9184	0.0672	0.0796	0.0097	24.4499	(1, 13, 15)

```
model = svm.SVC(decision_function_shape='ovo', kernel='rbf', C = 10, gamma=1)
```

Apéndice 7: K vecinos más cercanos con selección de características

#vecinos	Eficiencia	Int_ Eficiencia	Sensibilidad	Int_ Sensibilidad	Precisión	Int_ Precision	F-Score	Int_ F-Score	Error_ Prueba	Int_ Error	Tiempo ejecución (seg)
1	0.9693	0.0073	0.9699	0.0398	0.9687	0.0367	0.9689	0.0342	0.0307	0.0073	0.0821
2	0.9663	0.0036	0.9659	0.0494	0.9668	0.0344	0.9659	0.0381	0.0337	0.0036	0.0851
3	0.958	0.0026	0.9577	0.048	0.9585	0.046	0.9577	0.0435	0.042	0.0026	0.0871
4	0.9602	0.0092	0.9601	0.0501	0.9607	0.0474	0.9597	0.0424	0.0398	0.0092	0.0891
5	0.9563	0.0014	0.9561	0.0514	0.9567	0.0484	0.9559	0.0457	0.0437	0.0014	0.0891
6	0.9529	0.0089	0.9548	0.0527	0.9541	0.0516	0.9538	0.0466	0.0471	0.0089	0.0941
7	0.9554	0.0077	0.9558	0.0525	0.9571	0.047	0.956	0.046	0.0446	0.0077	0.0931
8	0.9416	0.0067	0.9429	0.0647	0.9428	0.0643	0.9419	0.0583	0.0584	0.0067	0.0961
9	0.9503	0.0076	0.9491	0.0551	0.9499	0.059	0.9487	0.0515	0.0497	0.0076	0.0971
10	0.9546	0.0033	0.9537	0.0524	0.9546	0.0541	0.9537	0.0498	0.0454	0.0033	0.0951
11	0.9481	0.0059	0.9477	0.0578	0.9477	0.0504	0.9472	0.0503	0.0519	0.0059	0.0961
12	0.9459	0.0083	0.9441	0.0698	0.9441	0.0582	0.9435	0.0605	0.0541	0.0083	0.1001
13	0.9485	0.0077	0.9473	0.0568	0.9491	0.0653	0.9477	0.0581	0.0515	0.0077	0.1021
14	0.9386	0.0055	0.9382	0.0674	0.9395	0.0656	0.9381	0.0618	0.0614	0.0055	0.1021
15	0.9386	0.0061	0.9389	0.0717	0.9378	0.0685	0.9376	0.0656	0.0614	0.0061	0.0991
16	0.9343	0.0074	0.9348	0.0785	0.9337	0.0641	0.9336	0.0682	0.0657	0.0074	0.1021
17	0.9412	0.009	0.9406	0.0668	0.9401	0.0671	0.9397	0.0627	0.0588	0.009	0.1001
18	0.9416	0.0063	0.9414	0.0665	0.9427	0.0589	0.9412	0.0569	0.0584	0.0063	0.1051
19	0.9351	0.0036	0.9371	0.0689	0.9365	0.064	0.936	0.0621	0.0649	0.0036	0.1021
20	0.9312	0.0114	0.9322	0.0801	0.9316	0.0709	0.931	0.0711	0.0688	0.0114	0.1041

Apéndice 8. Perceptrón multicapa con selección de características

N. de capas ocultas	Neuronas por capa	Eficiencia	Int_Eficiencia	Sensibilidad	Int_Sensibilidad	Precision	Int_ Precision	F-Score	Int_ F-Score	Error_ Prueba	Int_ Error	Tiempo de ejecución(seg)
1	**5**	0.9399	0.009	0.9386	0.0787	0.9393	0.0817	0.9387	0.0792	0.0601	0.009	12.116
	10	0.9464	0.0056	0.9484	0.06	0.9489	0.0613	0.9483	0.0586	0.0536	0.0056	12.6545
	15	0.9585	0.0032	0.9585	0.0518	0.9586	0.0527	0.9581	0.0484	0.0415	0.0032	13.8556
	20	0.9589	0.0081	0.9593	0.0446	0.9599	0.0508	0.9594	0.0455	0.0411	0.0081	14.9806
	25	0.958	0.0123	0.959	0.0507	0.9585	0.06	0.9581	0.0515	0.042	0.0123	16.2047
	30	0.9645	0.0047	0.9647	0.0508	0.9649	0.0466	0.9642	0.0431	0.0355	0.0047	16.5421
	35	0.9645	0.0019	0.9649	0.0432	0.9654	0.041	0.9648	0.0387	0.0355	0.0019	17.1296
	40	0.9632	0.0039	0.9632	0.0443	0.9626	0.0477	0.9625	0.0424	0.0368	0.0039	17.7902

	45	0.9628	0.0086	0.9627	0.048	0.963	0.0452	0.9625	0.0433	0.0372	0.0086	17.2897
	50	0.9671	0.0071	0.9674	0.0383	0.9669	0.0405	0.967	0.037	0.0329	0.0071	17.0095
2	5	0.9529	0.0077	0.9522	0.0557	0.9517	0.0576	0.9518	0.0554	0.0471	0.0077	15.7363
	10	0.958	0.0069	0.9572	0.0475	0.9576	0.0511	0.9573	0.0482	0.042	0.0069	15.0527
	15	0.965	0.0019	0.9654	0.0394	0.965	0.0473	0.9648	0.0392	0.035	0.0019	17.3888
	20	0.9684	0.0103	0.969	0.0363	0.9693	0.0391	0.969	0.0359	0.0316	0.0103	15.6582
	25	0.9676	0.01	0.9686	0.0361	0.9684	0.0427	0.9683	0.0375	0.0324	0.01	15.8154
	30	0.9732	0.0029	0.9727	0.0289	0.9742	0.0309	0.9733	0.027	0.0268	0.0029	14.7504
	35	0.9702	0.0043	0.97	0.0353	0.9701	0.0356	0.9699	0.0337	0.0298	0.0043	15.2689
	40	0.9728	0.0031	0.9718	0.0333	0.9725	0.0309	0.9719	0.0294	0.0272	0.0031	15.5081
	45	0.9676	0.0093	0.9671	0.0396	0.9677	0.0389	0.9671	0.036	0.0324	0.0093	14.8185
	50	0.9736	0.0039	0.9726	0.0348	0.9725	0.0331	0.9724	0.0324	0.0264	0.0039	16.3969

Apéndice 9. Random forest con selección de características

#arboles	Variablesx nodo	Eficiencia	Int_Eficiencia	Sensibilidad	Int_Sensibilidad	Precisión	Int_ Precision	F-Score	Int_ F-Score	Error_ Prueba	Int_ Error	Tiempo ejecución (seg)
5	2	0.9632	0.0075	0.9621	0.0465	0.9631	0.047	0.9624	0.0447	0.0368	0.0075	0.05
	5	0.9719	0.0066	0.9723	0.0329	0.9722	0.0323	0.9721	0.03	0.0281	0.0066	0.0701
	10	0.9663	0.0026	0.9666	0.0359	0.9666	0.0371	0.9664	0.0338	0.0337	0.0026	0.1061
	13	0.965	0.0082	0.966	0.0418	0.9664	0.0429	0.9658	0.0383	0.035	0.0082	0.1291
10	2	0.9702	0.0099	0.9698	0.0385	0.9706	0.036	0.9699	0.0332	0.0298	0.0099	0.0841
	5	0.9749	0.0065	0.9744	0.0334	0.9745	0.0266	0.9743	0.0276	0.0251	0.0065	0.1251
	10	0.9749	0.0047	0.976	0.0288	0.9757	0.0279	0.9756	0.0251	0.0251	0.0047	0.1992
	13	0.9697	0.0026	0.9696	0.0337	0.9696	0.0317	0.9694	0.0294	0.0303	0.0026	0.2462
20	2	0.9758	0.0065	0.976	0.0286	0.9754	0.0286	0.9756	0.0272	0.0242	0.0065	0.1541
	5	0.9814	0.0041	0.9811	0.0255	0.9803	0.0278	0.9805	0.0225	0.0186	0.0041	0.2402
	10	0.9771	0.0069	0.977	0.0285	0.9775	0.026	0.9771	0.0252	0.0229	0.0069	0.3793
	13	0.9736	0.0057	0.9732	0.0322	0.9734	0.0334	0.9731	0.0309	0.0264	0.0057	0.4784
50	2	0.971	0.0043	0.9715	0.0312	0.9712	0.0351	0.9712	0.0313	0.029	0.0043	0.3643
	5	0.9818	0.0029	0.9815	0.0245	0.9816	0.0196	0.9814	0.0205	0.0182	0.0029	0.5705
	10	0.9779	0.006	0.9776	0.0271	0.978	0.0232	0.9777	0.0234	0.0221	0.006	0.9429
	13	0.9719	0.0051	0.9718	0.0307	0.9722	0.0359	0.9716	0.0274	0.0281	0.0051	1.1721
100	2	0.9775	0.0049	0.9772	0.0257	0.9772	0.0311	0.977	0.0261	0.0225	0.0049	0.7107
	5	0.9836	0.0043	0.983	0.0242	0.9835	0.0257	0.983	0.0203	0.0164	0.0043	1.123
	10	0.9732	0.0079	0.9733	0.0314	0.9736	0.0328	0.9732	0.0287	0.0268	0.0079	1.8607

	13	0.9745	0.0026	0.974	0.0266	0.9744	0.0276	0.9741	0.026	0.0255	0.0026	2.3341

Apéndice 10. Máquinas de Soporte Vectorial con kernel lineal y con kernel RBF y con selección de características

Kernel	C	gamma	Eficiencia	Int_Eficiencia	Sensibilidad	Int_Sensibilidad	Precisión	Int_Precision	F-Score	Int_F-Score	Error_Prueba	Int_Error	%VS	Tiempo ejecución (seg)
linear	**0.001**	**0.000**	0.1246	0.0056	0.1429	0.3499	0.0178	0.0436	0.0316	0.0776	0.8754	0.0056	0.9997	1.2381
	0.010	**0.000**	0.5368	0.0357	0.5548	0.4462	0.5486	0.3926	0.4607	0.3853	0.4632	0.0357	0.9779	1.096
	0.100	**0.000**	0.8651	0.0253	0.8631	0.1377	0.8615	0.1231	0.8612	0.1273	0.1349	0.0253	0.7364	0.5195
	1.000	**0.000**	0.9234	0.01	0.9245	0.0995	0.9247	0.099	0.924	0.0978	0.0766	0.01	0.4226	0.3023
	10.000	**0.000**	0.9433	0.0051	0.944	0.0679	0.9436	0.0685	0.9434	0.0662	0.0567	0.0051	0.2434	0.2853
	100.000	**0.000**	0.955	0.005	0.955	0.0493	0.9562	0.0571	0.9552	0.0501	0.045	0.005	0.1728	0.3643
	200.000	**0.000**	0.9567	0.005	0.9575	0.045	0.9578	0.0674	0.9568	0.052	0.0433	0.005	0.1618	0.4354
	300.000	**0.000**	0.955	0.01	0.9555	0.0526	0.9567	0.0594	0.9559	0.0547	0.045	0.01	0.1516	0.4644
rbf	**0.001**	**0.001**	0.125	0.0043	0.1429	0.3499	0.0179	0.0438	0.0317	0.0778	0.875	0.0043	1	2.4522
		0.010	0.1773	0.0867	0.1948	0.3783	0.0487	0.1047	0.0741	0.1534	0.8227	0.0867	1	2.4552
		0.100	0.1237	0.0078	0.1429	0.3499	0.0177	0.0434	0.0314	0.0772	0.8763	0.0078	0.9993	2.4522
		1.000	0.128	0.0032	0.1429	0.3499	0.0183	0.0448	0.0324	0.0794	0.872	0.0032	1	2.4692
	0.010	**0.001**	0.1185	0.0055	0.1429	0.3499	0.0169	0.0415	0.0303	0.0742	0.8815	0.0055	1	2.4412
		0.010	0.1293	0.0014	0.1429	0.3499	0.0185	0.0453	0.0327	0.0801	0.8707	0.0014	1	2.4592
		0.100	0.1453	0.038	0.1662	0.3612	0.0546	0.1878	0.0616	0.1624	0.8547	0.038	1	2.4562
		1.000	0.5497	0.0425	0.5712	0.4668	0.532	0.4162	0.467	0.4061	0.4503	0.0425	0.9727	2.2871
	0.100	**0.001**	0.1215	0.0043	0.1429	0.3499	0.0174	0.0426	0.031	0.0759	0.8785	0.0043	0.9999	2.4492
		0.010	0.1929	0.0552	0.2157	0.3898	0.1357	0.2927	0.1154	0.233	0.8071	0.0552	1	2.4542
		0.100	0.7331	0.0329	0.7441	0.2579	0.7723	0.2044	0.7243	0.2115	0.2669	0.0329	0.9169	1.8317
		1.000	0.8897	0.0033	0.8969	0.1293	0.8951	0.1063	0.8931	0.1127	0.1103	0.0033	0.6643	0.8958
	1.000	**0.001**	0.1531	0.0405	0.1733	0.3715	0.0562	0.1884	0.0683	0.1844	0.8469	0.0405	0.9997	2.4642
		0.010	0.7569	0.0307	0.768	0.2503	0.8053	0.1932	0.751	0.2002	0.2431	0.0307	0.9079	1.7466
		0.100	0.8958	0.0144	0.8961	0.1172	0.8968	0.1143	0.8951	0.1126	0.1042	0.0144	0.6373	0.7787
		1.000	0.9347	0.0086	0.9362	0.0818	0.9366	0.0803	0.9349	0.0744	0.0653	0.0086	0.3444	0.4454
	10.000	**0.001**	0.7539	0.0097	0.7587	0.2426	0.7889	0.1801	0.7431	0.1784	0.2461	0.0097	0.9114	1.7456
		0.010	0.8953	0.0104	0.8971	0.1213	0.896	0.114	0.8963	0.1171	0.1047	0.0104	0.6328	0.7677
		0.100	0.9282	0.0076	0.9309	0.0858	0.9298	0.0852	0.9297	0.0833	0.0718	0.0076	0.3391	0.4164
		1.000	0.9615	0.011	0.9626	0.0435	0.9626	0.0522	0.9624	0.0461	0.0385	0.011	0.2028	0.3363
	100.000	**0.001**	0.8966	0.0028	0.8993	0.1137	0.8981	0.1041	0.8979	0.1069	0.1034	0.0028	0.6334	0.7717

		0.010	0.9356	0.0063	0.9347	0.0848	0.9347	0.0799	0.9344	0.0811	0.0644	0.0063	0.3405	0.4354
		0.100	0.9615	0.0014	0.961	0.0547	0.9609	0.0496	0.9605	0.0487	0.0385	0.0014	0.1966	0.3623
		1.000	0.9736	0.0031	0.9744	0.0329	0.9744	0.031	0.9741	0.0277	0.0264	0.0031	0.1572	0.3583
	200.000	**0.001**	0.9148	0.0052	0.9161	0.1048	0.9148	0.0999	0.9153	0.1017	0.0852	0.0052	0.5367	0.6256
		0.010	0.9477	0.0063	0.9451	0.0776	0.9452	0.0696	0.9447	0.0717	0.0523	0.0063	0.2994	0.4184
		0.100	0.9637	0.0076	0.9632	0.0489	0.9632	0.0422	0.9628	0.0423	0.0363	0.0076	0.176	0.3603
		1.000	0.9693	0.0077	0.9708	0.0361	0.9707	0.0357	0.9705	0.0329	0.0307	0.0077	0.149	0.3823
	300.000	**0.001**	0.9074	0.0058	0.9094	0.1148	0.9087	0.1116	0.9083	0.1111	0.0926	0.0058	0.4749	0.5545
		0.010	0.9334	0.0015	0.9348	0.0832	0.9359	0.0778	0.935	0.0791	0.0666	0.0015	0.2595	0.3914
		0.100	0.9693	0.0022	0.9696	0.034	0.9705	0.0408	0.9698	0.0351	0.0307	0.0022	0.1682	0.3603
		1.000	0.9732	0.0086	0.9731	0.03	0.9733	0.0344	0.9729	0.0289	0.0268	0.0086	0.1446	0.3924

Apéndice 11. K vecinos más cercanos con PCA:

KNeighborsClassifier(n_neighbors = 1)

Numero de Componentes	Eficiencia	Int_ Eficiencia	Sensibilidad	Int_Sensibilidad	Precision	Int_Precision	F-Score	Int_ F-Score	Error_Prueba	Int_ Error	Tiempo ejecución(seg)
1	0.5584	0.0096	0.5636	0.2086	0.5594	0.2037	0.5595	0.2048	0.4416	0.0096	0.6906
2	0.6466	0.0189	0.6493	0.1706	0.6493	0.1692	0.6478	0.1678	0.3534	0.0189	0.7387
3	0.8421	0.0028	0.8429	0.1244	0.8419	0.1215	0.8417	0.1214	0.1579	0.0028	0.7146
4	0.8932	0.006	0.8914	0.0935	0.892	0.0787	0.8907	0.0821	0.1068	0.006	0.7617
5	0.9161	0.0158	0.9171	0.0738	0.9169	0.0689	0.9159	0.0648	0.0839	0.0158	0.7136
6	0.9252	0.0095	0.9233	0.0731	0.9239	0.0603	0.9229	0.0624	0.0748	0.0095	0.7106
7	0.9386	0.0087	0.9387	0.0689	0.9388	0.0516	0.938	0.0552	0.0614	0.0087	0.7177
8	0.9386	0.0043	0.9387	0.0646	0.939	0.0574	0.9384	0.058	0.0614	0.0043	0.7397
9	0.9516	0.0102	0.9531	0.0561	0.9516	0.0439	0.9518	0.0462	0.0484	0.0102	0.7066
10	0.9477	0.0083	0.9493	0.0527	0.9497	0.0415	0.949	0.0421	0.0523	0.0083	0.7137
11	0.9593	0.007	0.9587	0.0495	0.9595	0.0372	0.9587	0.0391	0.0407	0.007	0.7197
12	0.9572	0.0037	0.9564	0.0533	0.9573	0.0405	0.9564	0.0427	0.0428	0.0037	0.7106
13	0.9585	0.0042	0.9592	0.0527	0.9588	0.0394	0.9583	0.0393	0.0415	0.0042	0.7357
14	0.9667	0.0041	0.9678	0.0423	0.9672	0.0316	0.967	0.0313	0.0333	0.0041	0.7767
15	0.9554	0.0039	0.9572	0.0494	0.9551	0.0462	0.9555	0.0426	0.0446	0.0039	0.7247
16	0.9606	0.0054	0.9596	0.0471	0.9599	0.0364	0.9595	0.0391	0.0394	0.0054	0.7507
17	0.9702	0.005	0.9687	0.0398	0.9697	0.0314	0.9689	0.0316	0.0298	0.005	0.7527
18	0.955	0.0094	0.9549	0.0576	0.9565	0.0442	0.9547	0.0432	0.045	0.0094	0.7487

19	0.9658	0.0026	0.965	0.0436	0.9649	0.035	0.9645	0.0336	0.0342	0.0026	0.7717

Apéndice 12. Perceptrón multicapa con PCA:

MLPClassifier(activation='tanh',max_iter = 1000,hidden_layer_sizes=(15,15))

#Componentes	Eficiencia	Int_ Eficiencia	Sensibilidad	Int_Sensibilidad	Precisión	Int_Precision	F-Score	Int_ F-Score	Error_ Prueba	Int_ error	Tiempo ejecución(seg)
1	0.5484	0.0072	0.5428	0.2909	0.5196	0.2521	0.5142	0.2737	0.4516	0.0072	9.6608
2	0.673	0.0074	0.6745	0.2003	0.6792	0.156	0.6688	0.1693	0.327	0.0074	20.4336
3	0.8287	0.0128	0.8292	0.1519	0.8309	0.1339	0.8271	0.1369	0.1713	0.0128	19.9472
4	0.8858	0.0064	0.8879	0.0974	0.8887	0.0798	0.8863	0.0799	0.1142	0.0064	21.1142
5	0.9057	0.0083	0.9035	0.1021	0.9052	0.0868	0.9029	0.0889	0.0943	0.0083	21.0822
6	0.9178	0.0078	0.9161	0.0941	0.917	0.0709	0.915	0.076	0.0822	0.0078	21.1212
7	0.9446	0.0096	0.9448	0.0557	0.9443	0.0477	0.9441	0.0478	0.0554	0.0096	20.7058
8	0.9403	0.0058	0.9386	0.0683	0.9386	0.0611	0.9381	0.0615	0.0597	0.0058	21.2293
9	0.9572	0.0031	0.9566	0.0474	0.957	0.0438	0.9565	0.043	0.0428	0.0031	20.1894
10	0.9585	0.0047	0.9579	0.0445	0.9577	0.0421	0.9575	0.0399	0.0415	0.0047	22.1802
11	0.965	0.0112	0.965	0.0425	0.9657	0.0414	0.9652	0.0405	0.035	0.0112	19.7209
12	0.9524	0.0036	0.9532	0.0512	0.9529	0.0494	0.9526	0.047	0.0476	0.0036	20.2484
13	0.9693	0.0071	0.9689	0.0368	0.9689	0.0392	0.9686	0.0349	0.0307	0.0071	18.6089
14	0.9684	0.0048	0.9689	0.0345	0.9691	0.0304	0.9689	0.0307	0.0316	0.0048	18.666
15	0.9715	0.0047	0.9712	0.0368	0.9709	0.0306	0.9709	0.0323	0.0285	0.0047	18.5959
16	0.9723	0.0072	0.9714	0.0352	0.9734	0.0328	0.9722	0.0316	0.0277	0.0072	18.4868
17	0.9697	0.0061	0.9699	0.0366	0.9701	0.0376	0.9698	0.0343	0.0303	0.0061	16.6351
18	0.9788	0.0063	0.9794	0.027	0.9785	0.0294	0.9788	0.0254	0.0212	0.0063	18.7441
19	0.9749	0.0036	0.9753	0.0282	0.9759	0.0271	0.9755	0.0248	0.0251	0.0036	19.5058

Apéndice 13. Random Forest con PCA:

RandomForestClassifier(n_estimators = 20, max_features = 10)

#Componentes	Eficiencia	Int_Eficiencia	Sensibilidad	Int_Sensibilidad	Precisión	Int_Precision	F-Score	Int_ F-Score	Error_ Prueba	Int_ error	Tiempo ejecución(seg)
10	0.9394	0.0044	0.9394	0.0585	0.9389	0.0517	0.9389	0.0535	0.0606	0.0044	1.3122
11	0.942	0.007	0.9398	0.0597	0.9407	0.0566	0.94	0.0563	0.058	0.007	1.3092
12	0.936	0.0047	0.937	0.0645	0.9367	0.052	0.9362	0.0537	0.064	0.0047	1.3012
13	0.9412	0.0138	0.9398	0.0646	0.9404	0.0654	0.9398	0.0632	0.0588	0.0138	1.2792
14	0.9459	0.0076	0.9468	0.054	0.9473	0.0581	0.9467	0.0536	0.0541	0.0076	1.2611
15	0.9507	0.0068	0.9502	0.0598	0.9508	0.0528	0.95	0.0529	0.0493	0.0068	1.3122
16	0.9438	0.0074	0.9451	0.0613	0.9448	0.0628	0.9444	0.0587	0.0562	0.0074	1.3252
17	0.9537	0.005	0.9549	0.0527	0.9544	0.0477	0.9541	0.0452	0.0463	0.005	1.2882
18	0.952	0.0051	0.9519	0.0519	0.9525	0.0486	0.952	0.0487	0.048	0.0051	1.1991

19	0.9715	0.0055	0.9714	0.0341	0.9718	0.0271	0.9714	0.0271	0.0285	0.0055	1.2922

Apéndice 14. Máquinas de Soporte Vectorial con kernel lineal y con kernel RBF y con PCA:

SVC(gamma=0.1, C=float(300), kernel='rbf', decision_function_shape='ovo' , probability = True)

#Componentes	Eficiencia	Int_ Eficiencia	Sensibilidad	Int_Sensibilidad	Precisión	Int_Precision	F-Score	Int_ F-Score	Error_ Prueba	Int_ error	%vs	Tiempo ejecución(seg)
1	0.5294	0.0115	0.5472	0.3313	0.5647	0.2037	0.4993	0.2616	0.4706	0.0115	0.7536	2.3392
2	0.6315	0.0113	0.6436	0.2371	0.6678	0.1756	0.6299	0.1827	0.3685	0.0113	0.6686	1.9898
3	0.8452	0.0146	0.8452	0.1539	0.8484	0.1171	0.8415	0.1267	0.1548	0.0146	0.3801	1.5794
4	0.8659	0.0062	0.8681	0.1365	0.8762	0.1011	0.8654	0.0998	0.1341	0.0062	0.3145	1.5884
5	0.9066	0.0087	0.9041	0.1071	0.9095	0.0809	0.9032	0.0802	0.0934	0.0087	0.303	1.6755
6	0.92	0.0085	0.9187	0.0925	0.9207	0.0727	0.9184	0.0765	0.08	0.0085	0.2679	1.5074
7	0.9446	0.0096	0.9434	0.0628	0.9436	0.0519	0.9431	0.0545	0.0554	0.0096	0.2561	1.5103
8	0.9459	0.0045	0.945	0.0591	0.945	0.0523	0.9444	0.0512	0.0541	0.0045	0.2386	1.3826
9	0.9516	0.007	0.9527	0.0576	0.9523	0.0446	0.9519	0.047	0.0484	0.007	0.2309	1.3462
10	0.9563	0.0109	0.9556	0.0457	0.9558	0.0456	0.9553	0.042	0.0437	0.0109	0.2423	1.3622
11	0.9654	0.0012	0.964	0.0416	0.9653	0.0402	0.9642	0.0356	0.0346	0.0012	0.2315	1.3622
12	0.9615	0.0059	0.9615	0.0478	0.961	0.0443	0.9609	0.0428	0.0385	0.0059	0.2285	1.3242
13	0.9628	0.0047	0.9638	0.0418	0.9632	0.0453	0.963	0.0389	0.0372	0.0047	0.2164	1.342
14	0.9676	0.0065	0.9684	0.0367	0.9678	0.0367	0.9679	0.0341	0.0324	0.0065	0.2156	1.3128
15	0.9537	0.01	0.9542	0.06	0.9547	0.0514	0.9537	0.0502	0.0463	0.01	0.2141	1.3311
16	0.9667	0.0088	0.9676	0.0417	0.9674	0.0416	0.9673	0.0394	0.0333	0.0088	0.2201	1.3181
17	0.9676	0.0043	0.9677	0.0416	0.967	0.0377	0.967	0.0356	0.0324	0.0043	0.2165	1.3428
18	0.9663	0.0078	0.9655	0.0432	0.9658	0.043	0.9653	0.0401	0.0337	0.0078	0.2159	1.3312
19	0.9702	0.0014	0.9708	0.0382	0.9707	0.0365	0.9704	0.0333	0.0298	0.0014	0.2168	1.4273