UNIVERSIDADE DA CORUÑA

Título
Clasificación de mangos mediante técnicas de aprendizaje

Resumen
En este trabajo se analiza el desempeño de varias técnicas de clasificación para categorizar mangos en tres clases diferentes de acuerdo con su integridad física. Los métodos seleccionados son Análisis discriminante lineal, K vecinos más cercanos, Bosques aleatorios y Redes neuronales de convolución. Entre las redes neuronales, además, se evalúan tres arquitecturas de entre las más utilizadas en el estado del arte para problemas de clasificación de imágenes: ResNet, MobileNet y Vgg16.

Autores
Daniel Vidal Soroa
Juan Diego Peña N
19/03/2022

Rev.	Descripción	Elaborado	Aprobado
01	Primera edición		

UNIVERSIDADE DA CORUÑA

Índice

1. Introducción

1.1. Objeto

El objetivo de este proyecto es implementar técnicas de aprendizaje automático supervisado sobre un conjunto de datos definido. Cada método se evaluará escogiendo las métricas de error más adecuadas, para luego optimizar sus hiperparámetros y determinar el de mayor rendimiento.

1.2. Alcance

El proyecto incluye la aplicabilidad de modelos ya existentes en las librerías scikit-learn [1] y keras [2]. Así como el uso datos de dominio público, el proyecto no contempla el uso de técnicas de aprendizaje no supervisado y se ceñirá únicamente a las técnicas: LDA, KNN, RF y CNN.

Normas y referencias

[1] «Scikit learn,» [En línea]. Available: https://scikit-learn.org/stable/.

[2] «TensorFlow,» [En línea]. Available: https://www.tensorflow.org/guide/keras?.

1.3. Definiciones y abreviaturas

ML Machine learning (*Aprendizaje automático)*
LDA Linear Discriminant Analysis (Análisis *discriminante lineal)*
KNN K-Nearest Neighbor (K-Vecinos)
RF *Random Forest (Bosques aleatorios)*
CNN Convolutional neural networks (Redes neuronales convolucionales)

2. Descripción general

El proyecto se motiva en servir como una herramienta base para países productores de mango al eliminar un control de calidad manual que posibilita el daño de la fruta y abarata el costo de otros métodos de control como el uso resonancias magnéticas. Además, debido a la migración rural negativa y el corto período de conservación de esta fruta, cada año se pierden millones de toneladas debido a falta de mano de obra y errores de clasificación.

Los datos se seleccionaron de la plataforma kaggle [3] y cuentan con tres clases de mango y 200 imágenes por clase:

- ***Clase extra:*** Usados generalmente para la exportación, no contienen defectos o son lo suficientemente leves para no afectar su aspecto, calidad o conservación en general.
- ***Clase I:*** Destinados al consumo local, de buena calidad con algunos defectos como quemaduras por el sol, rozaduras, etc.
- ***Clase II:*** Reservados normalmente para el procesamiento industrial, cuentan con los mismos defectos de la Clase I a una escala mayor.

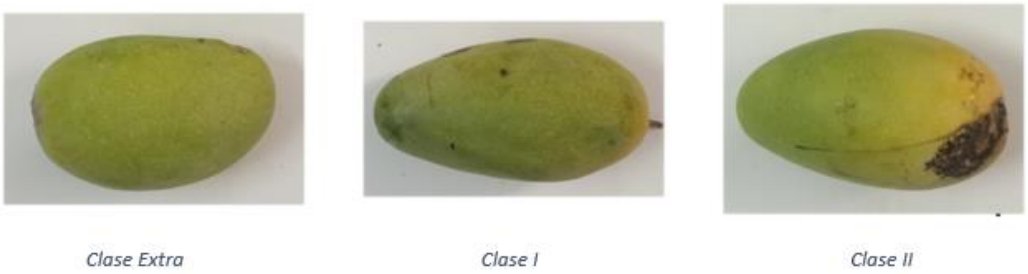


Figura 1. Clases de mango disponible

3. Diseño e implementación

El proceso de implementación de las técnicas de aprendizaje se basa en el propuesto en la guía de usuario de scikit-learn. Este se basa en un flujo de trabajo típico para este tipo de aplicaciones.

En primer lugar, se dividen los datos en dos grupos, uno de validación y otro de entrenamiento. Posteriormente, se optimizan los hiperparámetros de cada modelo para consecutivamente elegir el modelo más adecuado. Tanto la selección de hiperparámetros como de modelo se hace mediante una prueba estadística. Se muestra un diagrama del flujo en el que se basa el proyecto:

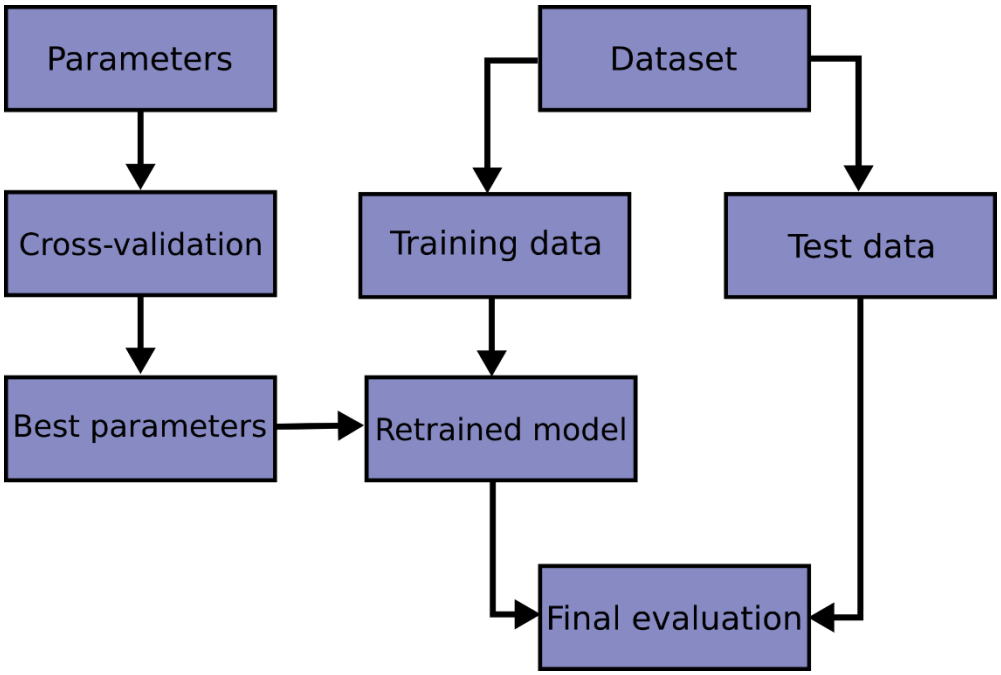


Figura 2. Flujo de trabajo típico

3.1. Preprocesamiento de datos

Primeramente, se definió un tamaño de imagen de 32x32 pixeles con el fin de reducir el tiempo de procesamiento y extraer de manera más eficiente características relevantes de las imágenes. Seguidamente, utilizando la librería de tensor Flow, se generaron 1800 imágenes nuevas para cada clase. Estas imágenes generadas se usan como entrenamiento y las 200 imágenes originales se dejan como test de validación. Los datos también fueron reorganizados aleatoriamente para que el orden no sea un factor determinante para los clasificadores.

Figura 3. Ejemplo de imágenes generadas y redimensionadas.

3.2. Evaluación de modelos y optimización de hiperparámetros

Las métricas utilizadas para la evaluación de todos los modelos serán, la exactitud, sensibilidad, precisión, f1-score, especificidad y área bajo la curva ROC. Con estas métricas y en conjunción con el método de validación cruzada se hará una evaluación objetiva de cada clasificador. Ya que el conjunto de datos es relativamente mediano (6000 imágenes), los datos de entrenamiento se dividirán en 5 grupos diferentes.

3.2.1. Análisis discriminante lineal (LDA)

Esta técnica se basa en encontrar el hiperplano que mejor separe los datos proporcionados. Ya que este método cuenta con pocos hiperparámetros, se realizó un análisis de los tipos de *solucionadores (solver):*

Parámetros	Parámetros propuestos
Solucionador	[svd, lsqr]

Tabla 1. Hiperparámetros analizados.

3.2.1.1. Resultado hiperparámetros

A continuación, se muestran los resultados obtenidos para cada métrica de error propuesta evaluando los dos tipos de solucionadores:

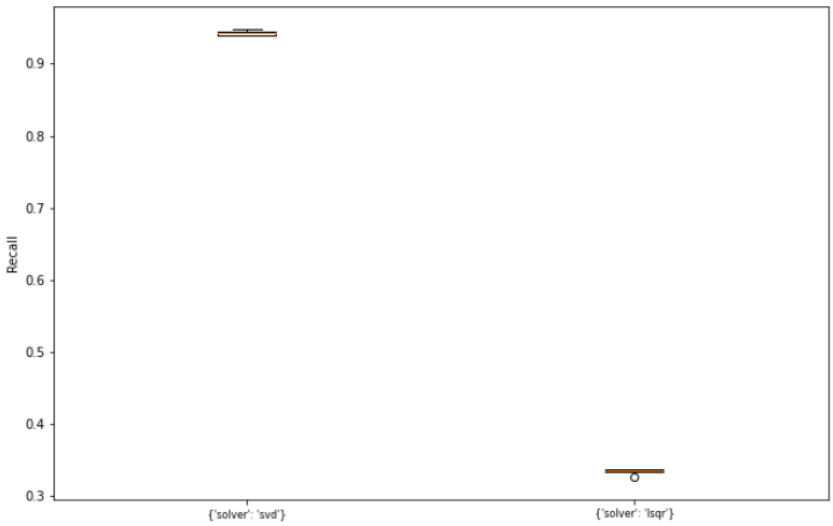


Figura 4. Resultados sensibilidad (recall).

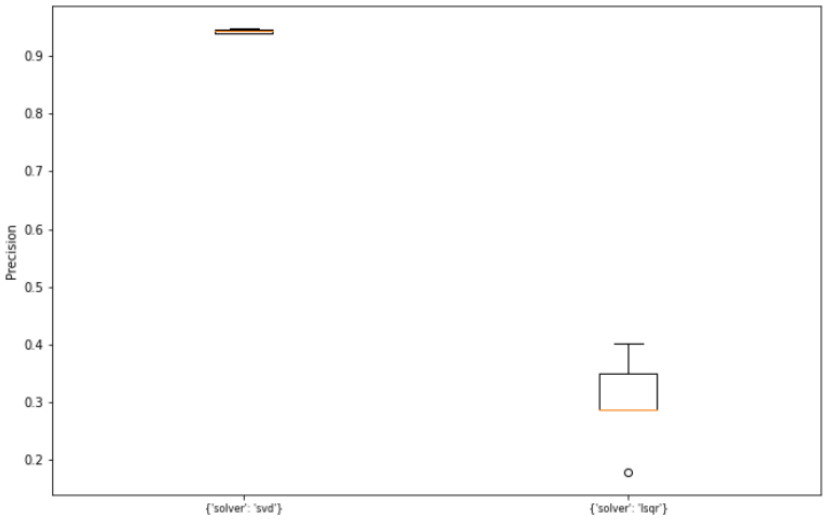


Figura 5. Resultados precisión.

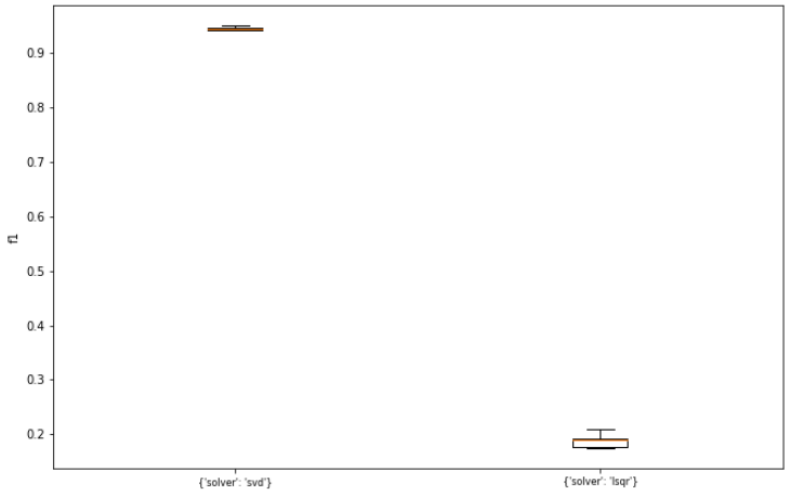


Figura 6. Resultados F1.

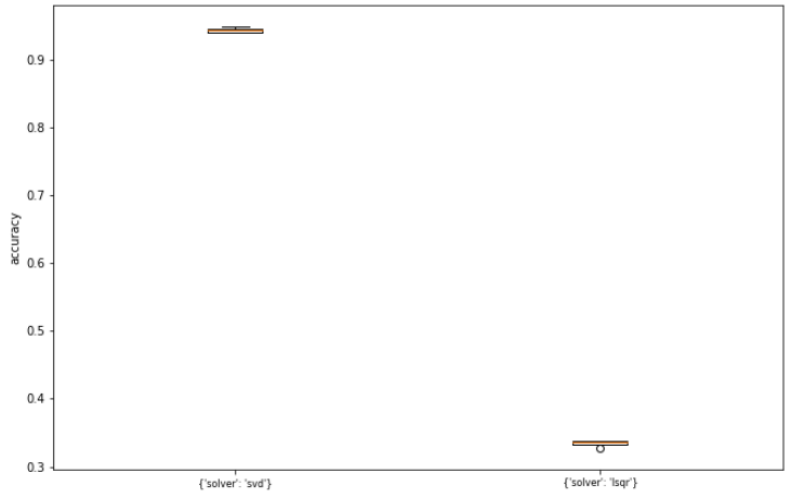


Figura 7 Resultados exactitud.

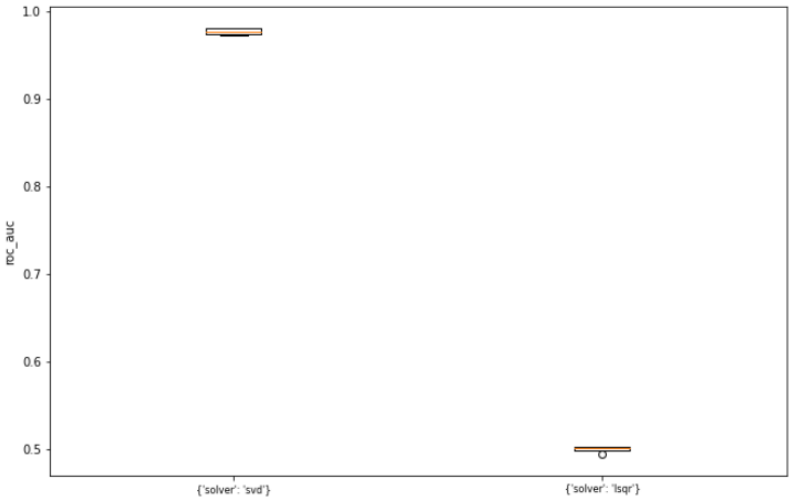


Figura 8. Resultados, área bajo la curva ROC.

Aunque en los resultados se evidencia un claro mejor desempeño por parte del solucionador *svd (descomposición de valores singulares),* se realizó una prueba estadística de Wilcoxon que comprueba que los modelos son diferentes estadísticamente, por lo que el entrenamiento final se realizó con el solucionador *svd.*

3.2.1.2. Evaluación final del modelo

Luego de determinar la mejor combinación de hiperparámetros, mediante validación cruzada, se hace una validación final con los datos dispuestos para este propósito. A continuación, los resultados:

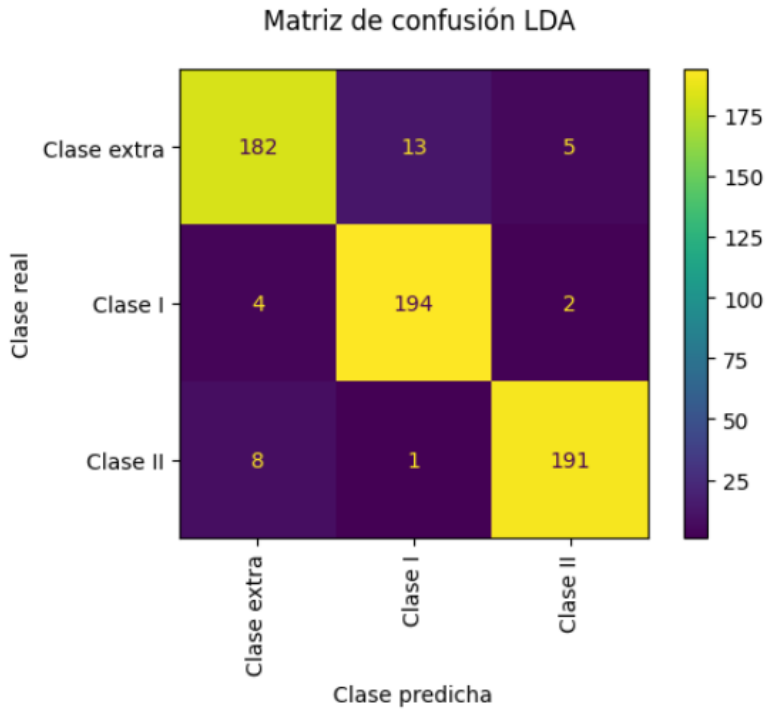


Figura 9. Matriz de confusión para el clasificador LDA.

Sensibilidad	**Precisión**	**F1**	**Exactitud**	**AUC**	**Especificidad**
0.945	0.945	0.945	0.945	0.972	0.972

Tabla 2. Resultados en datos de validación.

3.2.2. K-Vecinos (KNN)

El clasificador KNN se encarga de almacenar los datos de entrenamiento y realiza la clasificación basándose en una votación de vecinos. Cada vez que se realiza una predicción, se le asigna una clase a la entrada basado en la clase preasignada de los vecinos más cercanos.

Uno de los algoritmos usados por el clasificador para escoger el vecino más cercano es *KD Tree*, que usa como uno de sus parámetros el *Leaf size* y será tomado en cuenta para el análisis.

Otro hiperparámetro a tener en cuenta es el número de vecinos empleado para realizar la clasificación, tomando en cuenta la recomendación de que este número sea menor a la raíz cuadrada del número de datos, se definió un rango de vecinos a evaluar.

Finalmente se evalúa también el modelo con dos definiciones de distancia mínima diferente, la euclidiana y la de Minkowski. A continuación, se muestra u resumen de los parámetros utilizados en la optimización:

Parámetros	Parámetros propuestos
Leaf size	[1, 11, 21, 31, 41]
K-vecinos	[5, 13, 21, 29, 37]
Tipo de distancia[1]	[Manhattan, Euclidiana]

Tabla 3. Hiperparámetros a optimizar.

3.2.2.1. Resultados:

Ya que se tomaron en cuenta 250 combinaciones de hiperparámetros, se mostrará solamente los 5 mejores resultados de cada métrica, en el anexo I se muestran las gráficas completas.

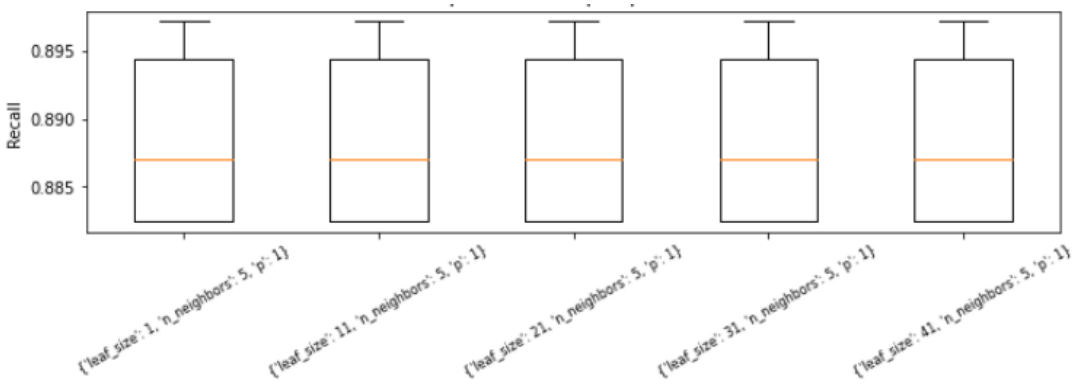


Figura 10. Resultados *sensibilidad*

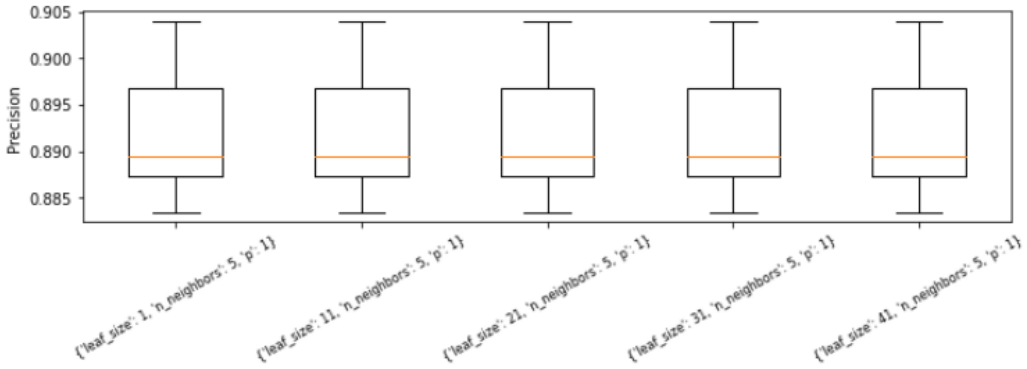


Figura 11. Resultados precisión.

[1] Representado en scikit-learn como p (1 =Manhattan 2 = Euclidiana)

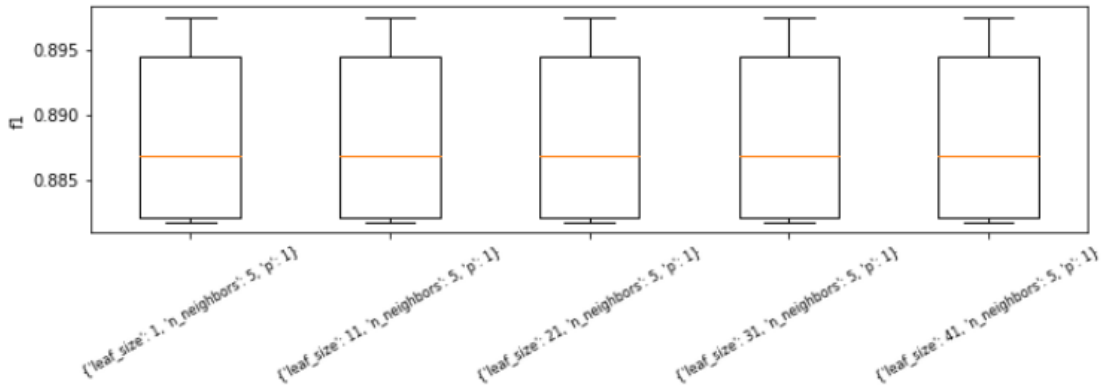


Figura 12. Resultados f1.

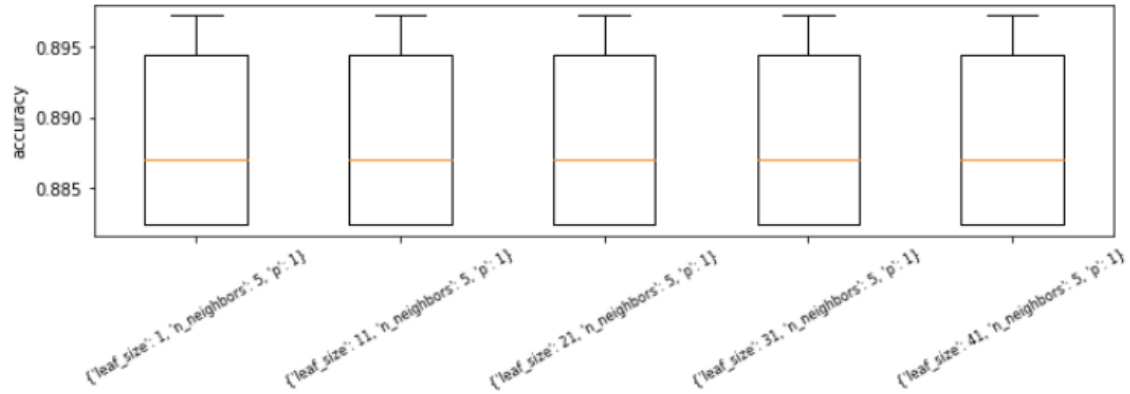


Figura 13. Resultados exactitud.

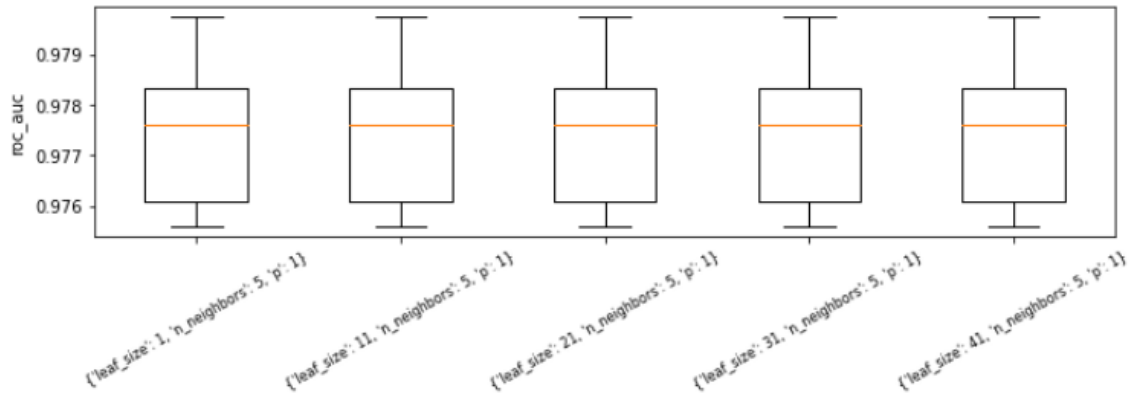


Figura 14. Resultados, área bajo la curva ROC.

Ya que el comportamiento de todas las métricas de error exhibe un comportamiento similar, se hace uso del *accuracy* o exactitud como métrica para realizar un análisis de hipótesis y decidir el mejor modelo. Cabe destacar por lo que la exactitud es un buen indicador para el análisis de hipótesis.

Usando una prueba de Wilcoxon, se compara la exactitud de cada una de las cinco divisiones de la validación cruzada. Dando como resultado que los modelos son iguales,

de manera que se entrena el modelo más eficiente computacionalmente con los siguientes parámetros:

Leaf size	*# Vecinos*	*Tipo de distancia*
41	5	Manhattan

Tabla 4. Parámetros finales del modelo.

3.2.2.2. Evaluación final del modelo

La prueba final del modelo se realiza con los datos de validación, a continuación, se muestran los resultados:

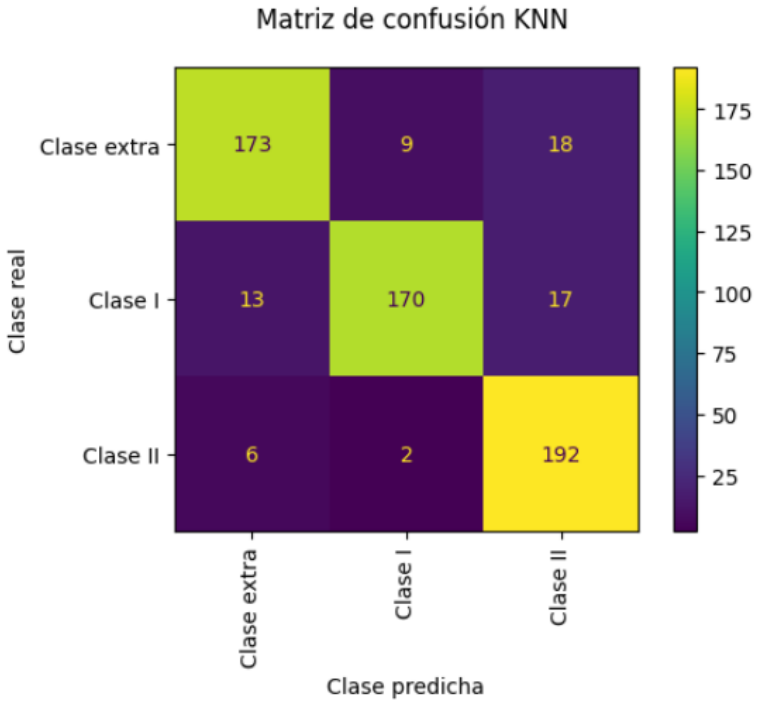


Figura 15. Matriz de confusión, modelo KNN.

Sensibilidad	Precisión	F1	Exactitud	AUC	Especificidad
0.892	0.895	0.891	0.892	0.980	0.946

Tabla 5. Métricas finales en datos de validación.

3.2.3. Bosque aleatorio (RF)

Random forest o bosque aleatorio, realiza la clasificación a través de una serie de pruebas a cada variable hasta alcanzar su clasificación, en cada prueba el conjunto de datos se divide en un cierto número, la cantidad de divisiones son controladas por el hiperparámetro *min_samples_split* el cual se usará para el análisis.

El número máximo de variables que se evalúa en cada prueba es otro hiperparámetro escogido para el análisis, así como el *boostrap* y *mean_samples_leaf*

Parámetros	Parámetros propuestos
Mínimo de divisiones	[2,5,10]
Mínimo de hojas	[1,2,4]
Bootstrap	[Falso, Verdadero]
Máximo de variables	[Automático, raíz cuadrada]

Tabla 6. Combinación de hiperparámetros a evaluar.

3.2.3.1. Resultados

Ya que se entrenaron mas de 180 combinaciones, se muestran solo los mejores dos resultados de cada métrica, los resultados completos se encuentran en el anexo I

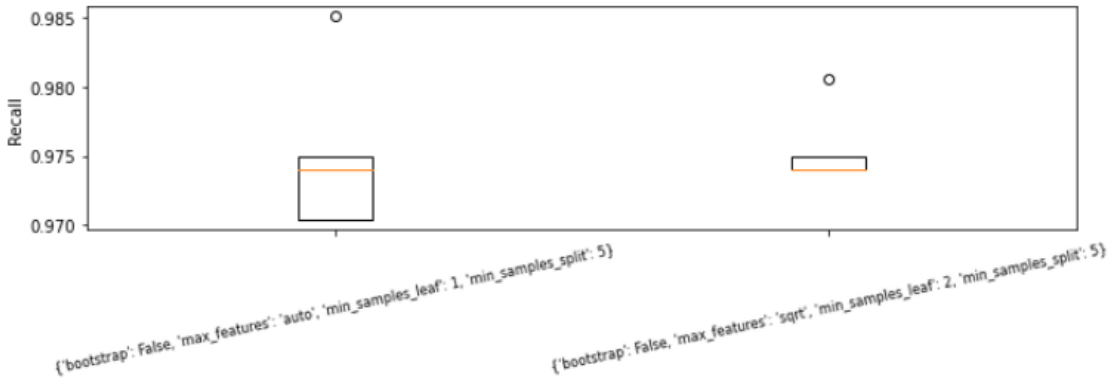


Figura 16. Resultados de sensibilidad.

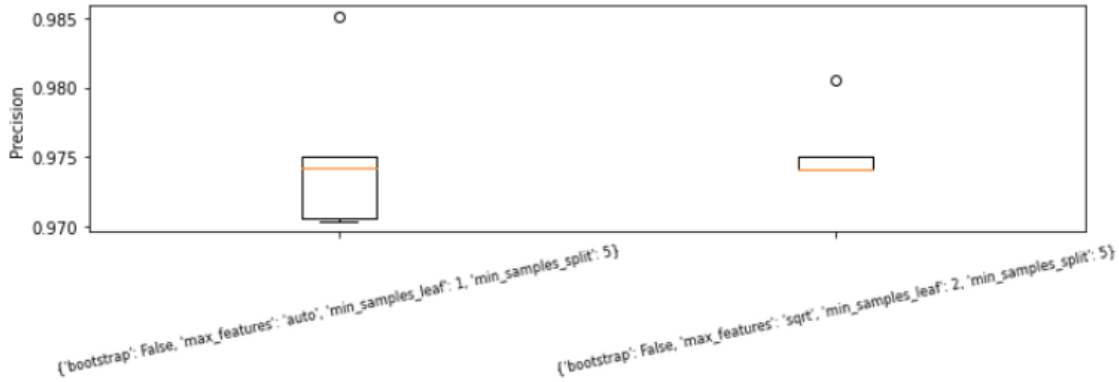


Figura 17. Resultados precisión.

Figura 18. Resultados f1.

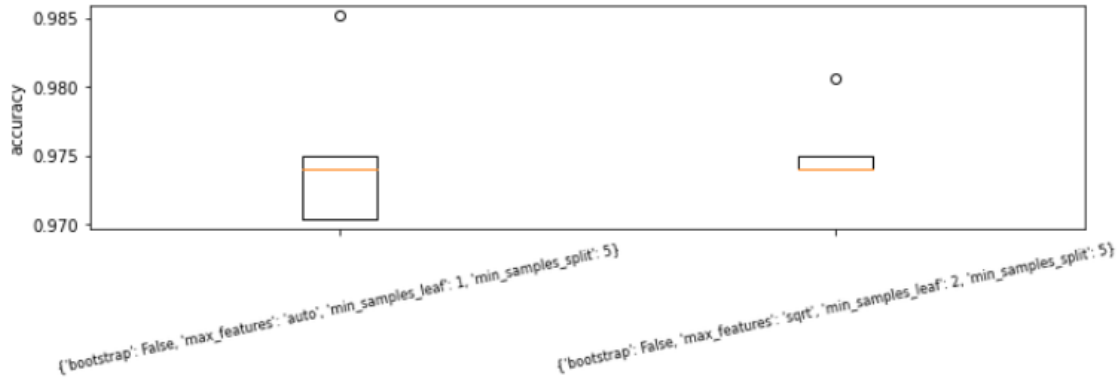


Figura 19. Resultados exactitud.

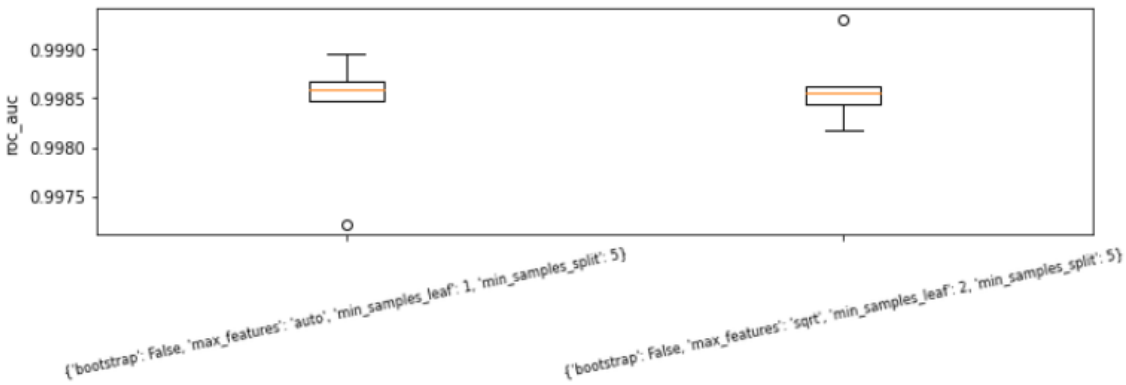


Figura 20. Resultados área bajo la curva ROC.

Aplicando de nuevo la prueba estadística de Wilcoxon, se llega al resultado que los modelos son iguales, por lo que se usa los parámetros que agilicen la velocidad de cómputo:

Boostrap	***Máximo de variables***	***Mínimo de hojas***	***Mínimo de divisiones***
Falso	Automático	2	5

Tabla 7. Modelo final.

3.2.3.2. *Evaluación final del modelo*

Luego de escoger el mejor modelo, se evalúa el desempeño con los datos de validación:

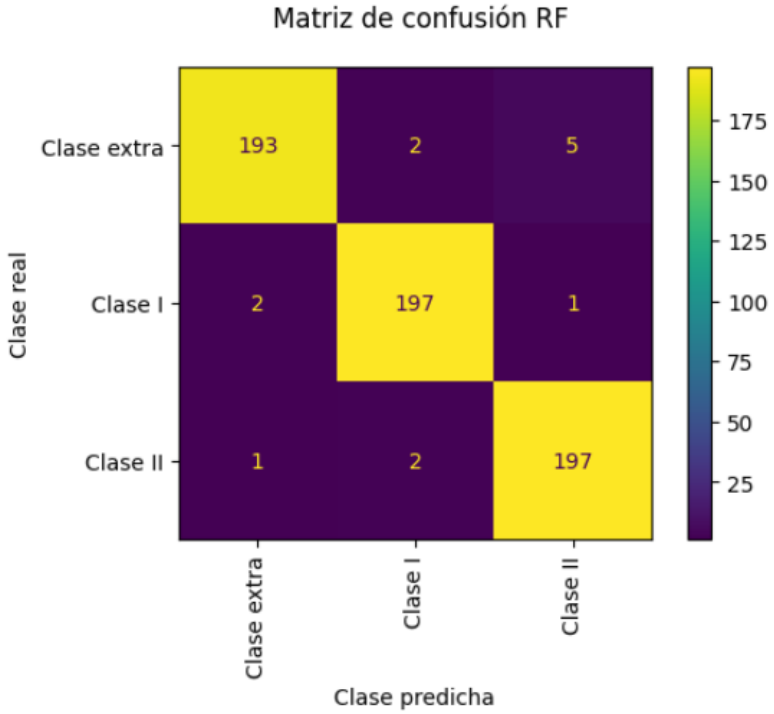


Figura 21. Matriz de confusión con los datos de validación.

Sensibilidad	**Precisión**	**F1**	**Exactitud**	**AUC**	**Especificidad**
0.982	0.982	0.982	0.982	0.999	0.991

Tabla 8. Métricas en datos de validación.

3.2.4. Redes neuronales convencionales (CNN)

Los clasificadores de redes artificiales convolucionales se basan en la aplicación de filtros sucesivos a imágenes, que facilitan la extracción de características que serán usadas luego para realizar la clasificación. Sin embargo, el desempeño de estas técnicas depende en gran medida de la calidad del proceso de entrenamiento. Específicamente, se recomienda contar con un conjunto numeroso de muestras de entrenamiento.

En este caso, incluso luego de haber aumentado el conjunto de datos original, no se disponían de datos suficientes para realizar un entrenamiento de calidad. Por este motivo, se decidió utilizar un concepto denominado transferencia de conocimiento, a través del cual se toma una red neuronal pre-entrenada, y se sustituyen sus capas finales por unas adecuadas al problema de clasificación en cuestión.

Para este trabajo, se tomaron en cuenta diferentes arquitecturas predefinidas en la librería keras. Los criterios seguidos para la selección de estas arquitecturas fueron la velocidad de cómputo, ya que esta aplicación se espera pueda desarrollarse en tiempo real, y la exactitud de la clasificación. La Tabla 9 muestra los modelos seleccionados y sus distintas configuraciones en cuanto a su profundidad y número de parámetros (pesos) totales.

Modelo	Exactitud	Parámetros	Profundidad	Tiempo (ms) por paso
Vgg16	90.1	138.4 M	16	69.5
ResNet50	92.1	25.6 M	107	58.2
MobileNet	89.5	4.3 M	55	22.6

Tabla 9: Modelos CNN seleccionados

3.2.4.1. Resultados:

Para la optimización de estas arquitecturas, se utilizaron tres combinaciones distintas de capas de salida para analizar cuáles presentan mejores resultados a la hora de transformar el conocimiento provisto por los modelos pre-entrenados en la clasificación final. Las combinaciones propuestas son como se muestran a continuación:

	Modelo 1	Modelo 2	Modelo 3
Capa 1	Aplanamiento	Aplanamiento	Aplanamiento
Capa 2	Densa Relu (64 neuronas)	Densa Relu (100 neuronas)	Densa Relu (128 neuronas)
Capa 3	Densa Softmax (3 neuronas)	Dropout (0.2)	Dropout (0.5)
Capa 4		Densa Softmax (3 neuronas)	Normalización de batch
Capa 5			Densa Softmax (3 neuronas)

Tabla 10: Arquitecturas propuestas para capas de salida

A continuación, se muestran algunos de los mejores resultados obtenidos combinando los tres modelos pre-entrenados con las configuraciones de salida propuestas. Como es posible apreciar, los mejores resultados se obtuvieron con la red Vgg16. Además, la arquitectura de salida más eficiente fue la más sencilla, compuesta por una capa densa de 64 neuronas y función de activación Relu y capa de salida de tres neuronas con activador Softmax.

Modelo base	Modelo final	Param	Epoch	Batch size	Exact.	Spec.	f1_scr	AUC
vgg16	1	33027	200	60	0,95	0,98	0,95	0,98
vgg16	1	33027	250	60	0,94	0,97	0,94	0,99
vgg16	2	66051	300	50	0,92	0,96	0,92	0,98
vgg16	3	52003	100	50	0,84	0,93	0,84	0,97

Tabla 11: Mejores resultados obtenidos

Modelo base	Modelo final	Param	Epoch	Batch size	Exact.	Spec.	f1_scr	AUC
mobilnet	1	65795	200	60	0,69	0,89	0,68	0,87
mobilnet	2	131587	250	60	0,68	0,88	0,65	0,86
resnet	1	131331	300	50	0,66	0,9	0,63	0,85
resnet	1	131331	200	60	0,63	0,91	0,56	0,82
resnet	1	131331	200	60	0,63	0,89	0,6	0,82

Tabla 12: Resultados obtenidos con modelos ResNet y MobileNet

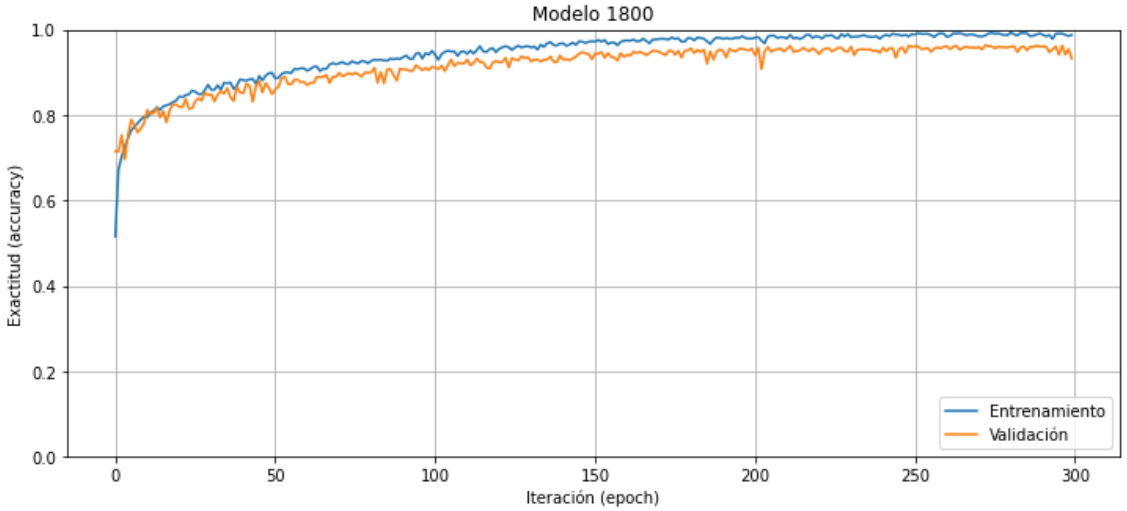


Figura 22. Arquitectura VGG16.

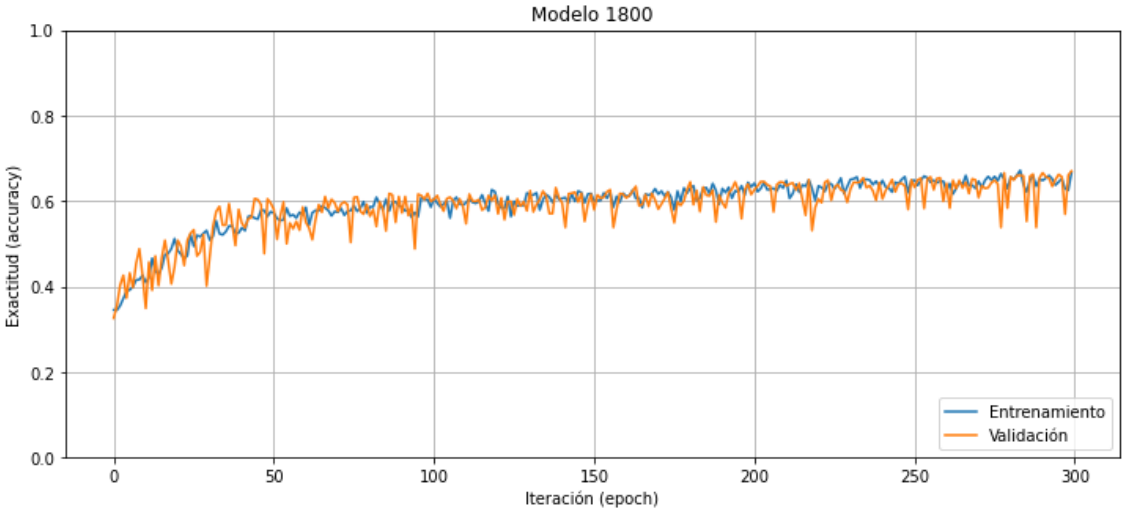


Figura 23. Arquitectura ResNet50

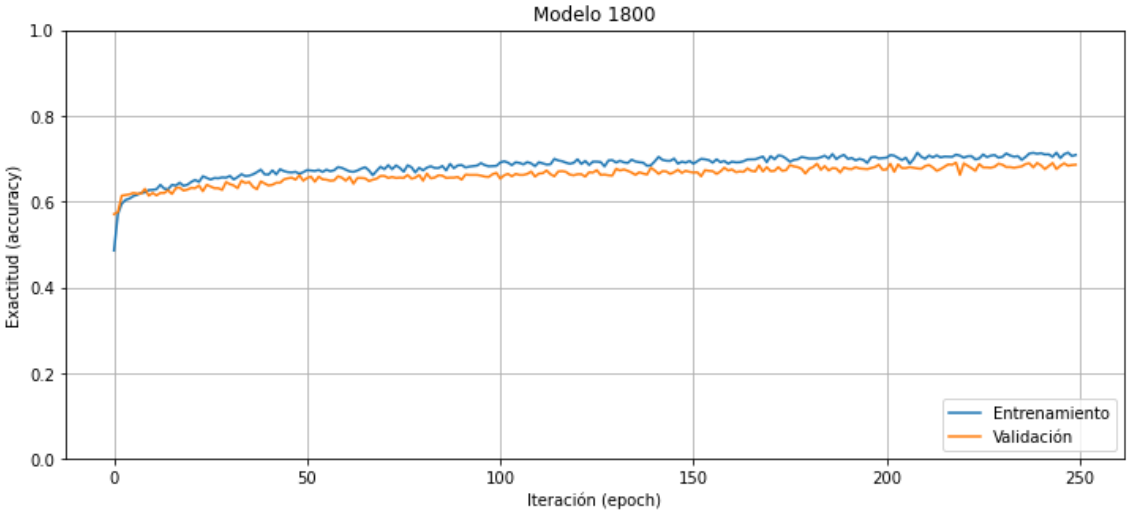


Figura 24: Arquitectura MobileNet

3.2.4.2. Evaluación final del modelo

A continuación, se muestran las métricas finales de la arquitectura elegida (Vgg16 + modelo de salida 1) utilizando los datos de prueba:

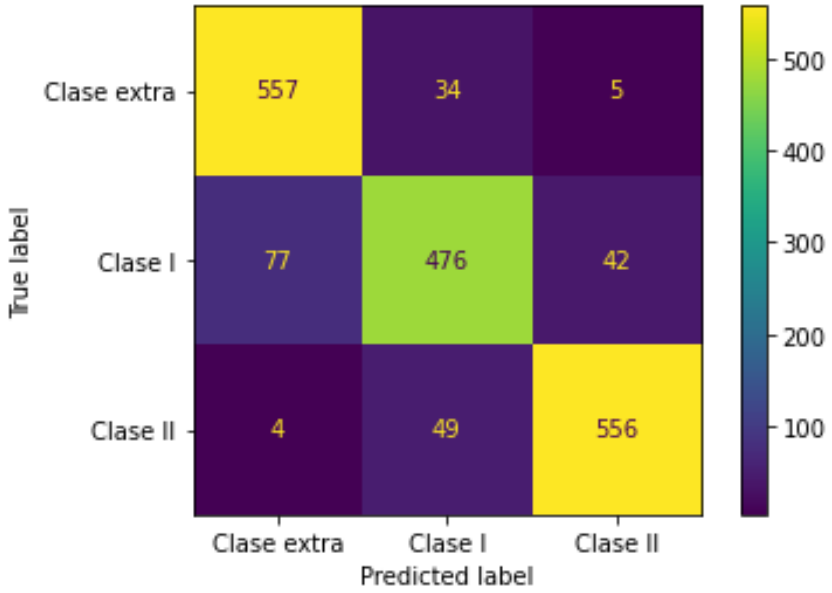


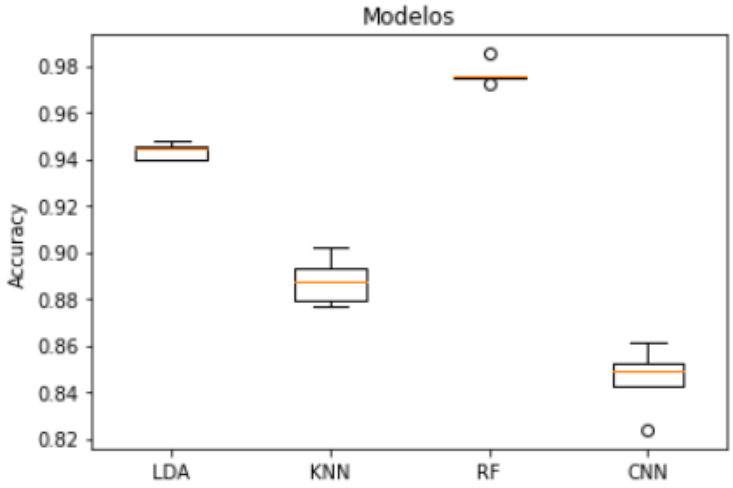
Figura 25. Matriz de confusión modelo CNN

Sensibilidad	**Precisión**	**F1**	**Exactitud**	**AUC**	**Especificidad**
0.95	0.95	0.95	0.95	0.98	0.98

Tabla 13. Métricas finales modelo CNN.

3.3. Comparación de modelos y selección

Luego de optimizar cada clasificador, se debe seleccionar el mejor. La métrica encargada realizar la comparación es la exactitud. Por cada modelo se realiza una prueba estadística para validar si los datos son iguales o no.



Luego de comprar los cuatro modelos con el contraste de hipótesis de Kruskall-Wallis con un Alpha de 0.05 se determino que los modelos son distintos. De manera que el modelo con mayor exactitud es el ***random forest***.

4. Conclusiones

El objetivo de este trabajo era seleccionar un algoritmo de clasificación que fuese capaz de discriminar entre tres clases de mango de acuerdo con su presencia física. Para ello, se han evaluado los clasificadores LDA, KNN, RF y CNN. Luego de ajustar los hiperparámetros de cada uno de estos métodos, se llega a la conclusión de que Random Forest es el que mejor resultados ofrece.

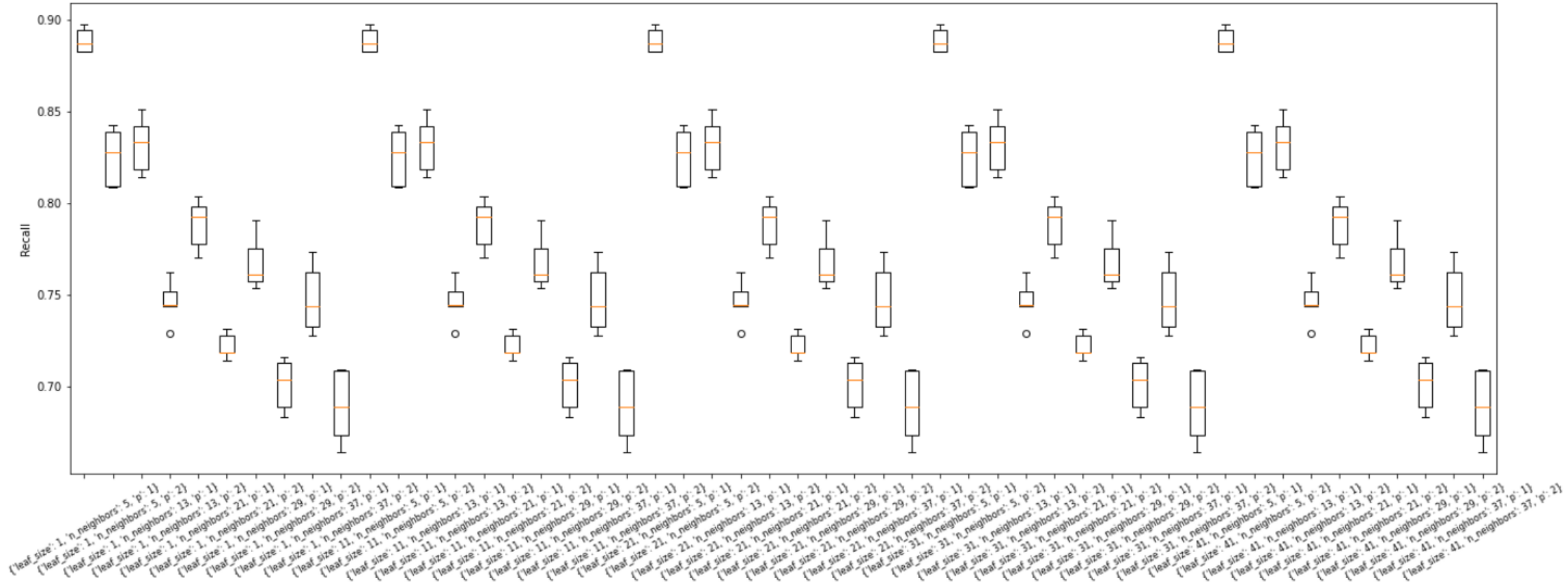
Este modelo es el más exacto y, además es un algoritmo relativamente rápido y fácil de parametrizar. Así mismo, la repetitividad con la que este algoritmo brindó buenos resultados también es destacable. Además, analizando la matriz de confusión obtenida durante la etapa de prueba, es posible apreciar que el clasificador es especialmente bueno distinguiendo entre las tres clases y, sobre todo, confunde muy pocas veces la clase 2 y la clase extra.

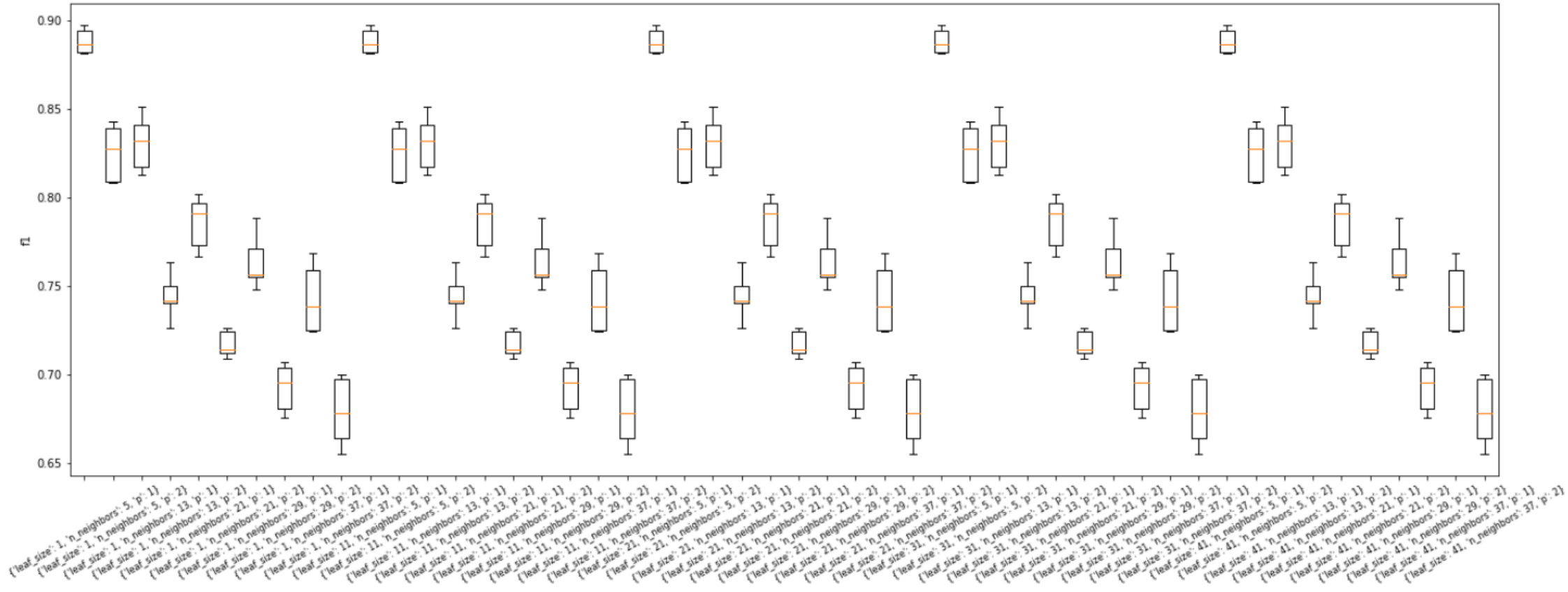
Esto es importante para la comercialización, ya que significa que muy pocos mangos no adecuados serán dedicados a la exportación, y muy pocas frutas premium serán enviadas a su procesamiento industrial.

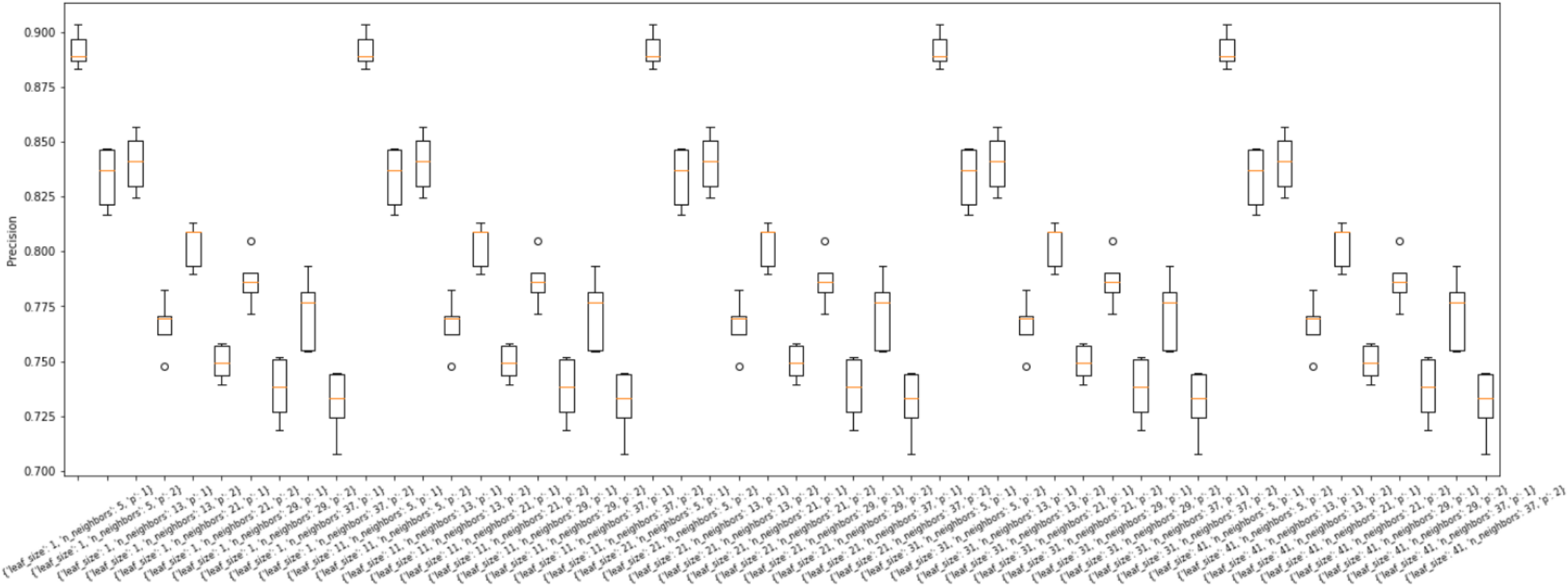
Respecto a las redes CNN, ha de decirse que mostraron mucho potencial, especialmente la red creada a partir de la arquitectura Vgg16. Durante su evaluación, las redes mostraron que podían mejorar su desempeño a partir de un aumento de las iteraciones (epochs), pudiendo llegar a cotas comparables incluso con RF. Sin embargo, para esta aplicación se prefiere el método antes mencionado, ya que este requiere menos parametrización y entrenamiento.

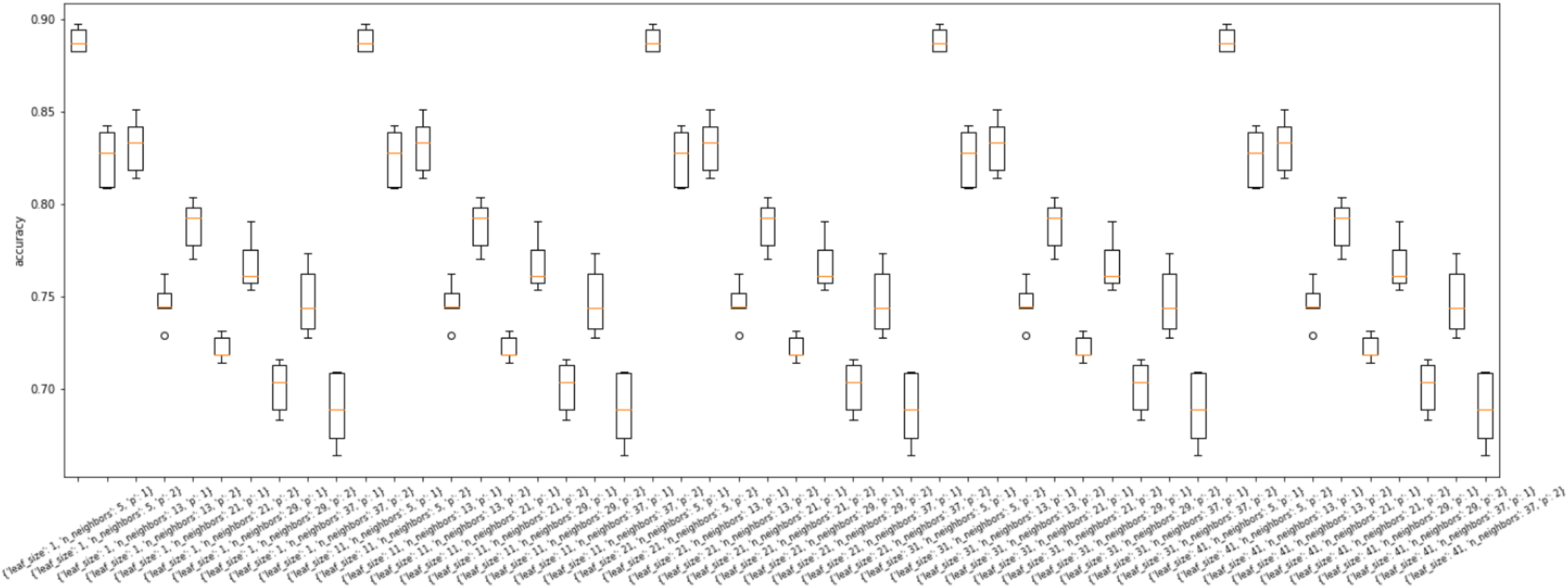
5. ANEXOS:

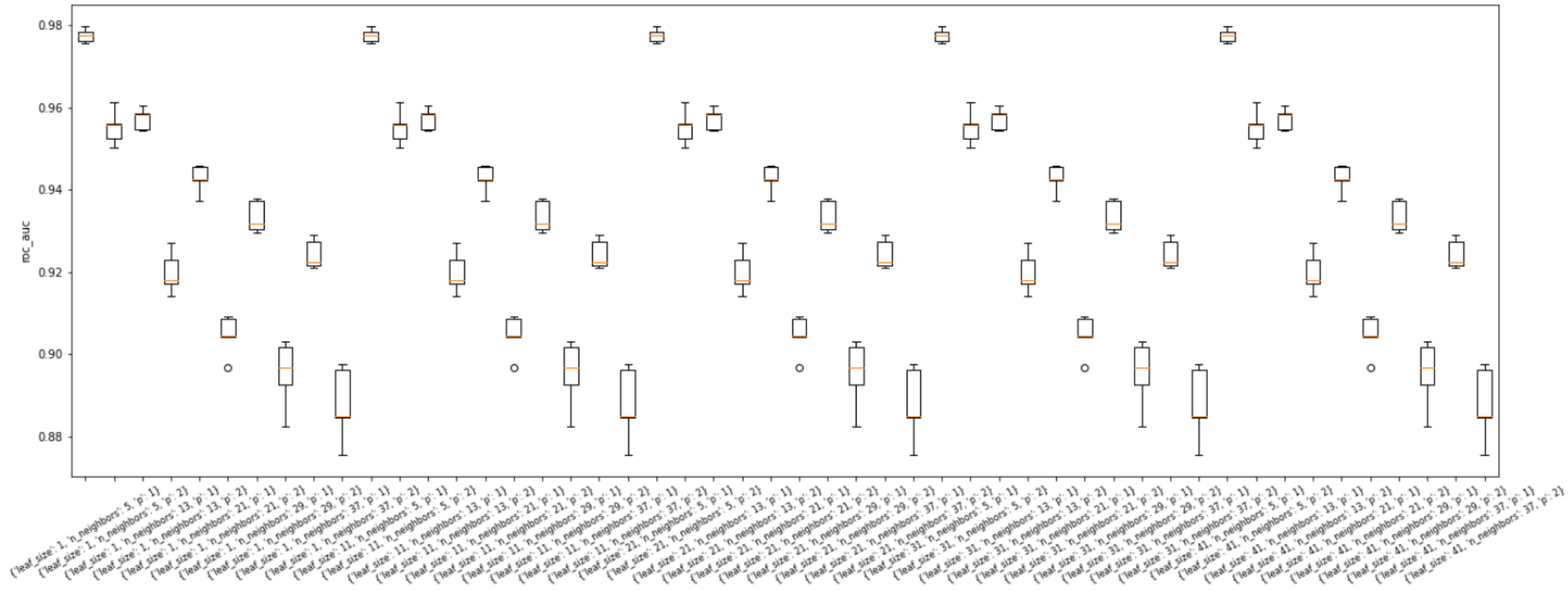
5.1. ANEXO I : ANALISIS DE HIPER PARAMETROS K-NN



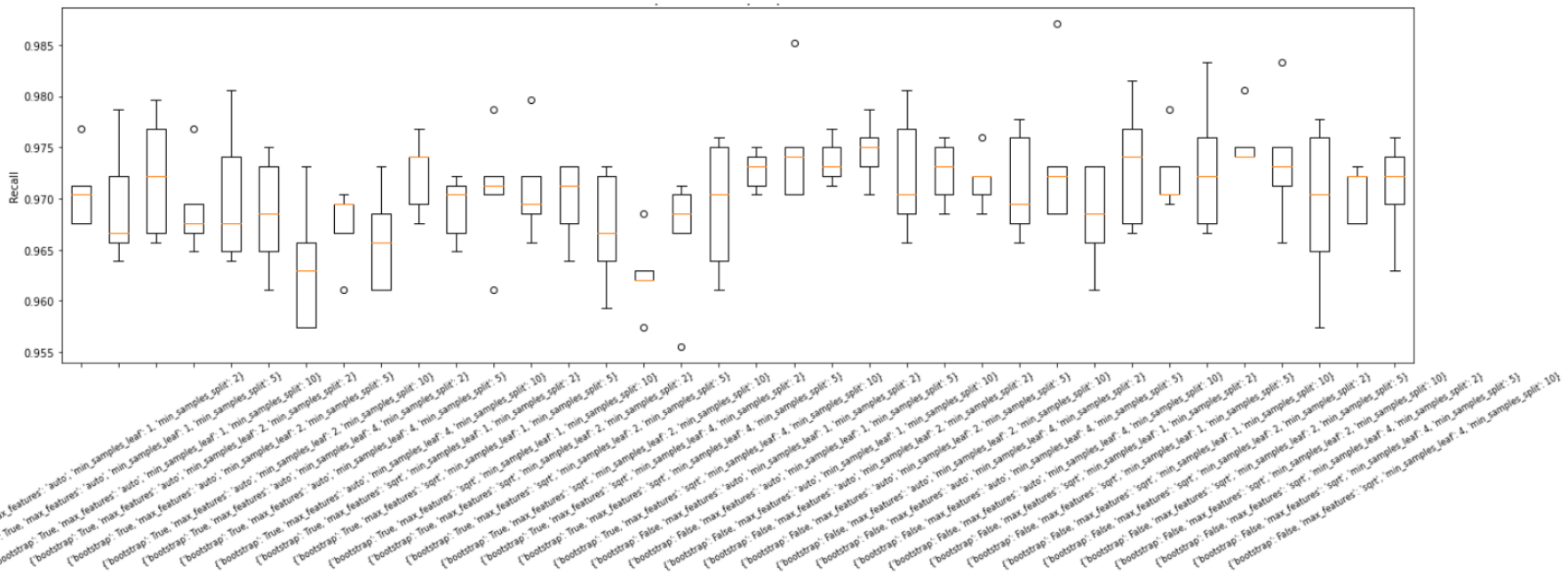

f1
0.90
0.85
0.80
0.75
0.70
0.65
{'leaf_size': 1, 'n_neighbors': 5, 'p': 1}
{'leaf_size': 1, 'n_neighbors': 5, 'p': 2}
{'leaf_size': 1, 'n_neighbors': 13, 'p': 1}
{'leaf_size': 1, 'n_neighbors': 13, 'p': 2}
{'leaf_size': 1, 'n_neighbors': 21, 'p': 1}
{'leaf_size': 1, 'n_neighbors': 21, 'p': 2}
{'leaf_size': 1, 'n_neighbors': 29, 'p': 1}
{'leaf_size': 1, 'n_neighbors': 29, 'p': 2}
{'leaf_size': 1, 'n_neighbors': 37, 'p': 1}
{'leaf_size': 1, 'n_neighbors': 37, 'p': 2}
{'leaf_size': 11, 'n_neighbors': 5, 'p': 1}
{'leaf_size': 11, 'n_neighbors': 5, 'p': 2}
{'leaf_size': 11, 'n_neighbors': 13, 'p': 1}
{'leaf_size': 11, 'n_neighbors': 13, 'p': 2}
{'leaf_size': 11, 'n_neighbors': 21, 'p': 1}
{'leaf_size': 11, 'n_neighbors': 21, 'p': 2}
{'leaf_size': 11, 'n_neighbors': 29, 'p': 1}
{'leaf_size': 11, 'n_neighbors': 29, 'p': 2}
{'leaf_size': 11, 'n_neighbors': 37, 'p': 1}
{'leaf_size': 11, 'n_neighbors': 37, 'p': 2}
{'leaf_size': 21, 'n_neighbors': 5, 'p': 1}
{'leaf_size': 21, 'n_neighbors': 5, 'p': 2}
{'leaf_size': 21, 'n_neighbors': 13, 'p': 1}
{'leaf_size': 21, 'n_neighbors': 13, 'p': 2}
{'leaf_size': 21, 'n_neighbors': 21, 'p': 1}
{'leaf_size': 21, 'n_neighbors': 21, 'p': 2}
{'leaf_size': 21, 'n_neighbors': 29, 'p': 1}
{'leaf_size': 21, 'n_neighbors': 29, 'p': 2}
{'leaf_size': 21, 'n_neighbors': 37, 'p': 1}
{'leaf_size': 21, 'n_neighbors': 37, 'p': 2}
{'leaf_size': 31, 'n_neighbors': 5, 'p': 1}
{'leaf_size': 31, 'n_neighbors': 5, 'p': 2}
{'leaf_size': 31, 'n_neighbors': 13, 'p': 1}
{'leaf_size': 31, 'n_neighbors': 13, 'p': 2}
{'leaf_size': 31, 'n_neighbors': 21, 'p': 1}
{'leaf_size': 31, 'n_neighbors': 21, 'p': 2}
{'leaf_size': 31, 'n_neighbors': 29, 'p': 1}
{'leaf_size': 31, 'n_neighbors': 29, 'p': 2}
{'leaf_size': 31, 'n_neighbors': 37, 'p': 1}
{'leaf_size': 31, 'n_neighbors': 37, 'p': 2}
{'leaf_size': 41, 'n_neighbors': 5, 'p': 1}
{'leaf_size': 41, 'n_neighbors': 5, 'p': 2}
{'leaf_size': 41, 'n_neighbors': 13, 'p': 1}
{'leaf_size': 41, 'n_neighbors': 13, 'p': 2}
{'leaf_size': 41, 'n_neighbors': 21, 'p': 1}
{'leaf_size': 41, 'n_neighbors': 21, 'p': 2}
{'leaf_size': 41, 'n_neighbors': 29, 'p': 1}
{'leaf_size': 41, 'n_neighbors': 29, 'p': 2}
{'leaf_size': 41, 'n_neighbors': 37, 'p': 1}
{'leaf_size': 41, 'n_neighbors': 37, 'p': 2}

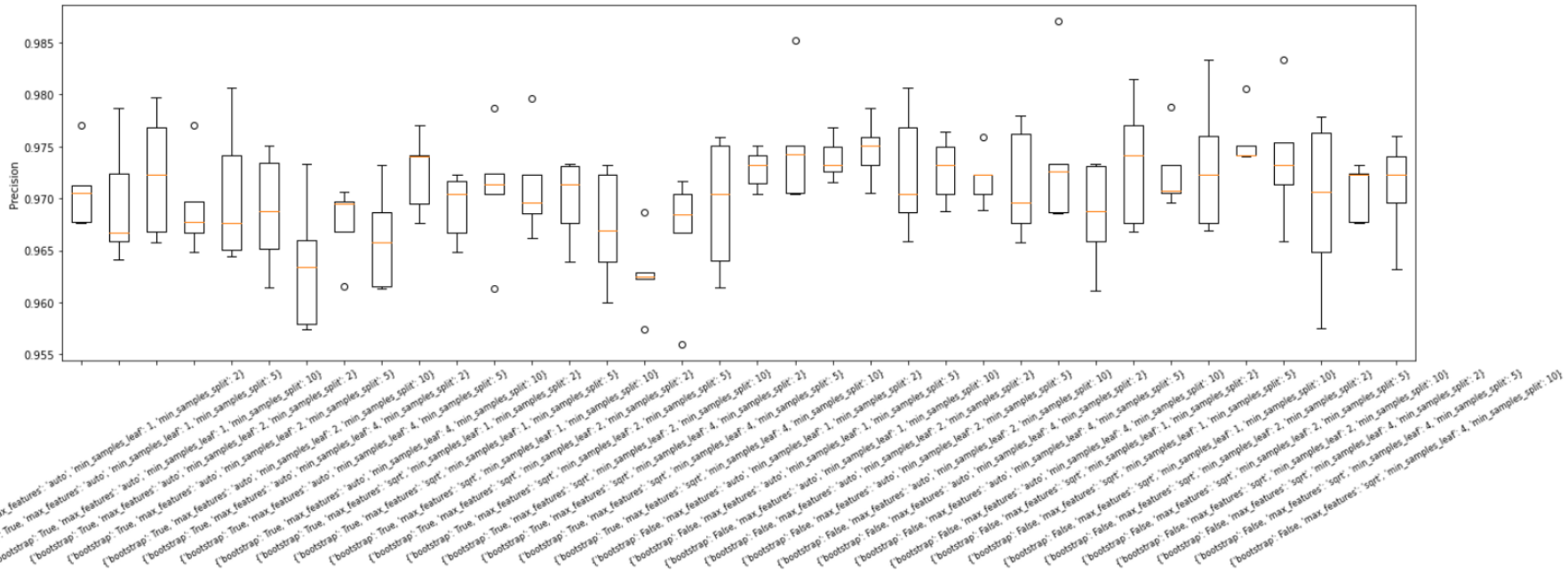

Precision
0.900
0.875
0.850
0.825
0.800
0.775
0.750
0.725
0.700
{'leaf_size': 1, 'n_neighbors': 5, 'p': 1}
{'leaf_size': 1, 'n_neighbors': 5, 'p': 2}
{'leaf_size': 1, 'n_neighbors': 13, 'p': 1}
{'leaf_size': 1, 'n_neighbors': 13, 'p': 2}
{'leaf_size': 1, 'n_neighbors': 21, 'p': 1}
{'leaf_size': 1, 'n_neighbors': 21, 'p': 2}
{'leaf_size': 1, 'n_neighbors': 29, 'p': 1}
{'leaf_size': 1, 'n_neighbors': 29, 'p': 2}
{'leaf_size': 1, 'n_neighbors': 37, 'p': 1}
{'leaf_size': 1, 'n_neighbors': 37, 'p': 2}
{'leaf_size': 11, 'n_neighbors': 5, 'p': 1}
{'leaf_size': 11, 'n_neighbors': 5, 'p': 2}
{'leaf_size': 11, 'n_neighbors': 13, 'p': 1}
{'leaf_size': 11, 'n_neighbors': 13, 'p': 2}
{'leaf_size': 11, 'n_neighbors': 21, 'p': 1}
{'leaf_size': 11, 'n_neighbors': 21, 'p': 2}
{'leaf_size': 11, 'n_neighbors': 29, 'p': 1}
{'leaf_size': 11, 'n_neighbors': 29, 'p': 2}
{'leaf_size': 11, 'n_neighbors': 37, 'p': 1}
{'leaf_size': 11, 'n_neighbors': 37, 'p': 2}
{'leaf_size': 21, 'n_neighbors': 5, 'p': 1}
{'leaf_size': 21, 'n_neighbors': 5, 'p': 2}
{'leaf_size': 21, 'n_neighbors': 13, 'p': 1}
{'leaf_size': 21, 'n_neighbors': 13, 'p': 2}
{'leaf_size': 21, 'n_neighbors': 21, 'p': 1}
{'leaf_size': 21, 'n_neighbors': 21, 'p': 2}
{'leaf_size': 21, 'n_neighbors': 29, 'p': 1}
{'leaf_size': 21, 'n_neighbors': 29, 'p': 2}
{'leaf_size': 21, 'n_neighbors': 37, 'p': 1}
{'leaf_size': 21, 'n_neighbors': 37, 'p': 2}
{'leaf_size': 31, 'n_neighbors': 5, 'p': 1}
{'leaf_size': 31, 'n_neighbors': 5, 'p': 2}
{'leaf_size': 31, 'n_neighbors': 13, 'p': 1}
{'leaf_size': 31, 'n_neighbors': 13, 'p': 2}
{'leaf_size': 31, 'n_neighbors': 21, 'p': 1}
{'leaf_size': 31, 'n_neighbors': 21, 'p': 2}
{'leaf_size': 31, 'n_neighbors': 29, 'p': 1}
{'leaf_size': 31, 'n_neighbors': 29, 'p': 2}
{'leaf_size': 31, 'n_neighbors': 37, 'p': 1}
{'leaf_size': 31, 'n_neighbors': 37, 'p': 2}
{'leaf_size': 41, 'n_neighbors': 5, 'p': 1}
{'leaf_size': 41, 'n_neighbors': 5, 'p': 2}
{'leaf_size': 41, 'n_neighbors': 13, 'p': 1}
{'leaf_size': 41, 'n_neighbors': 13, 'p': 2}
{'leaf_size': 41, 'n_neighbors': 21, 'p': 1}
{'leaf_size': 41, 'n_neighbors': 21, 'p': 2}
{'leaf_size': 41, 'n_neighbors': 29, 'p': 1}
{'leaf_size': 41, 'n_neighbors': 29, 'p': 2}
{'leaf_size': 41, 'n_neighbors': 37, 'p': 1}
{'leaf_size': 41, 'n_neighbors': 37, 'p': 2}

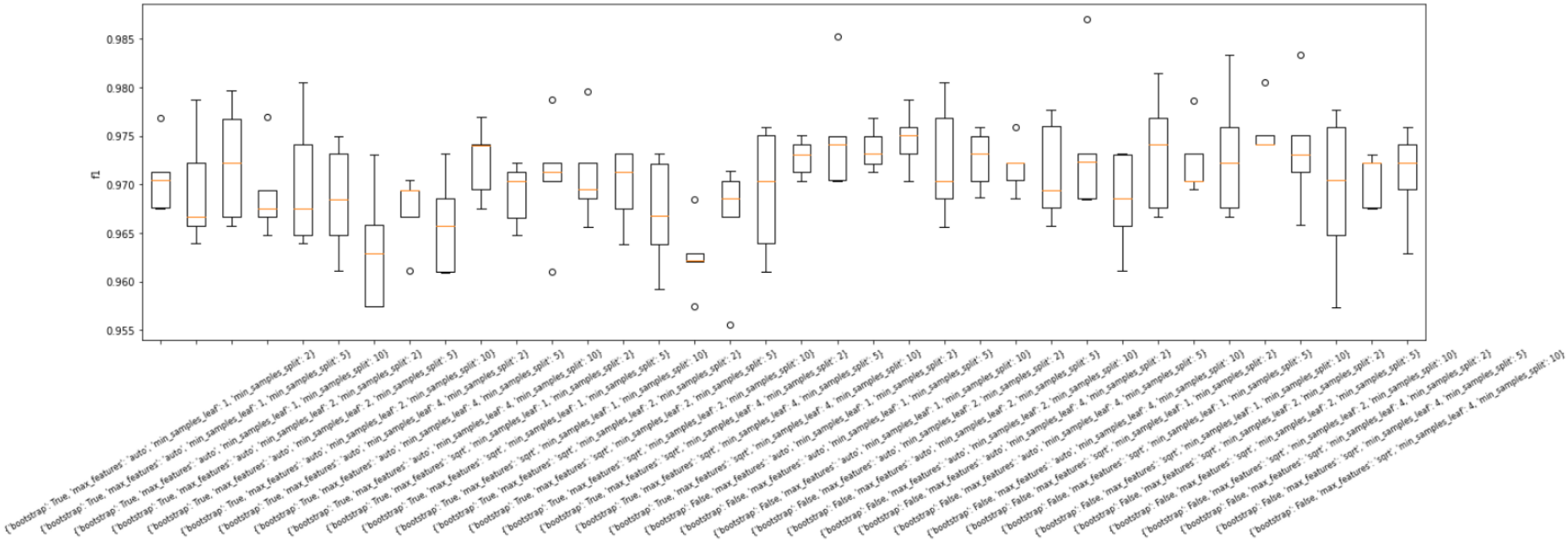

accuracy
0.90
0.85
0.80
0.75
0.70
{'leaf_size': 1, 'n_neighbors': 5, 'p': 1}
{'leaf_size': 1, 'n_neighbors': 5, 'p': 2}
{'leaf_size': 1, 'n_neighbors': 13, 'p': 1}
{'leaf_size': 1, 'n_neighbors': 13, 'p': 2}
{'leaf_size': 1, 'n_neighbors': 21, 'p': 1}
{'leaf_size': 1, 'n_neighbors': 21, 'p': 2}
{'leaf_size': 1, 'n_neighbors': 29, 'p': 1}
{'leaf_size': 1, 'n_neighbors': 29, 'p': 2}
{'leaf_size': 1, 'n_neighbors': 37, 'p': 1}
{'leaf_size': 1, 'n_neighbors': 37, 'p': 2}
{'leaf_size': 11, 'n_neighbors': 5, 'p': 1}
{'leaf_size': 11, 'n_neighbors': 5, 'p': 2}
{'leaf_size': 11, 'n_neighbors': 13, 'p': 1}
{'leaf_size': 11, 'n_neighbors': 13, 'p': 2}
{'leaf_size': 11, 'n_neighbors': 21, 'p': 1}
{'leaf_size': 11, 'n_neighbors': 21, 'p': 2}
{'leaf_size': 11, 'n_neighbors': 29, 'p': 1}
{'leaf_size': 11, 'n_neighbors': 29, 'p': 2}
{'leaf_size': 11, 'n_neighbors': 37, 'p': 1}
{'leaf_size': 11, 'n_neighbors': 37, 'p': 2}
{'leaf_size': 21, 'n_neighbors': 5, 'p': 1}
{'leaf_size': 21, 'n_neighbors': 5, 'p': 2}
{'leaf_size': 21, 'n_neighbors': 13, 'p': 1}
{'leaf_size': 21, 'n_neighbors': 13, 'p': 2}
{'leaf_size': 21, 'n_neighbors': 21, 'p': 1}
{'leaf_size': 21, 'n_neighbors': 21, 'p': 2}
{'leaf_size': 21, 'n_neighbors': 29, 'p': 1}
{'leaf_size': 21, 'n_neighbors': 29, 'p': 2}
{'leaf_size': 21, 'n_neighbors': 37, 'p': 1}
{'leaf_size': 21, 'n_neighbors': 37, 'p': 2}
{'leaf_size': 31, 'n_neighbors': 5, 'p': 1}
{'leaf_size': 31, 'n_neighbors': 5, 'p': 2}
{'leaf_size': 31, 'n_neighbors': 13, 'p': 1}
{'leaf_size': 31, 'n_neighbors': 13, 'p': 2}
{'leaf_size': 31, 'n_neighbors': 21, 'p': 1}
{'leaf_size': 31, 'n_neighbors': 21, 'p': 2}
{'leaf_size': 31, 'n_neighbors': 29, 'p': 1}
{'leaf_size': 31, 'n_neighbors': 29, 'p': 2}
{'leaf_size': 31, 'n_neighbors': 37, 'p': 1}
{'leaf_size': 31, 'n_neighbors': 37, 'p': 2}
{'leaf_size': 41, 'n_neighbors': 5, 'p': 1}
{'leaf_size': 41, 'n_neighbors': 5, 'p': 2}
{'leaf_size': 41, 'n_neighbors': 13, 'p': 1}
{'leaf_size': 41, 'n_neighbors': 13, 'p': 2}
{'leaf_size': 41, 'n_neighbors': 21, 'p': 1}
{'leaf_size': 41, 'n_neighbors': 21, 'p': 2}
{'leaf_size': 41, 'n_neighbors': 29, 'p': 1}
{'leaf_size': 41, 'n_neighbors': 29, 'p': 2}
{'leaf_size': 41, 'n_neighbors': 37, 'p': 1}
{'leaf_size': 41, 'n_neighbors': 37, 'p': 2}

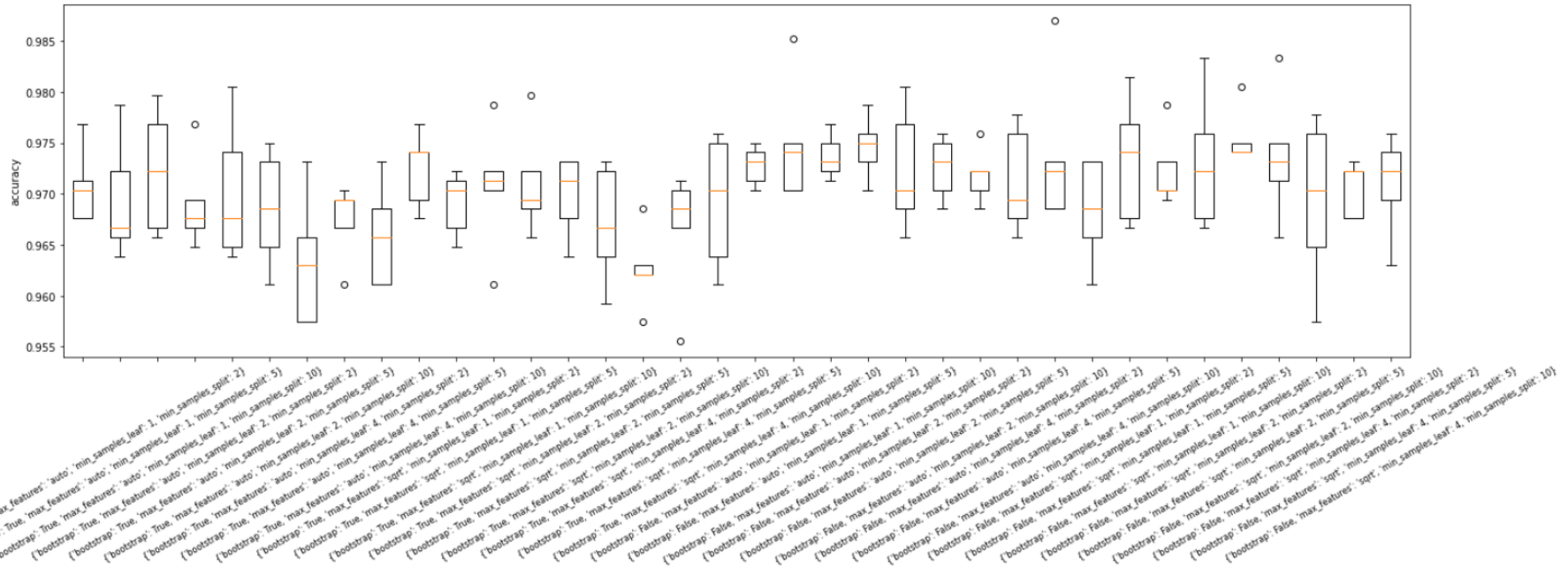


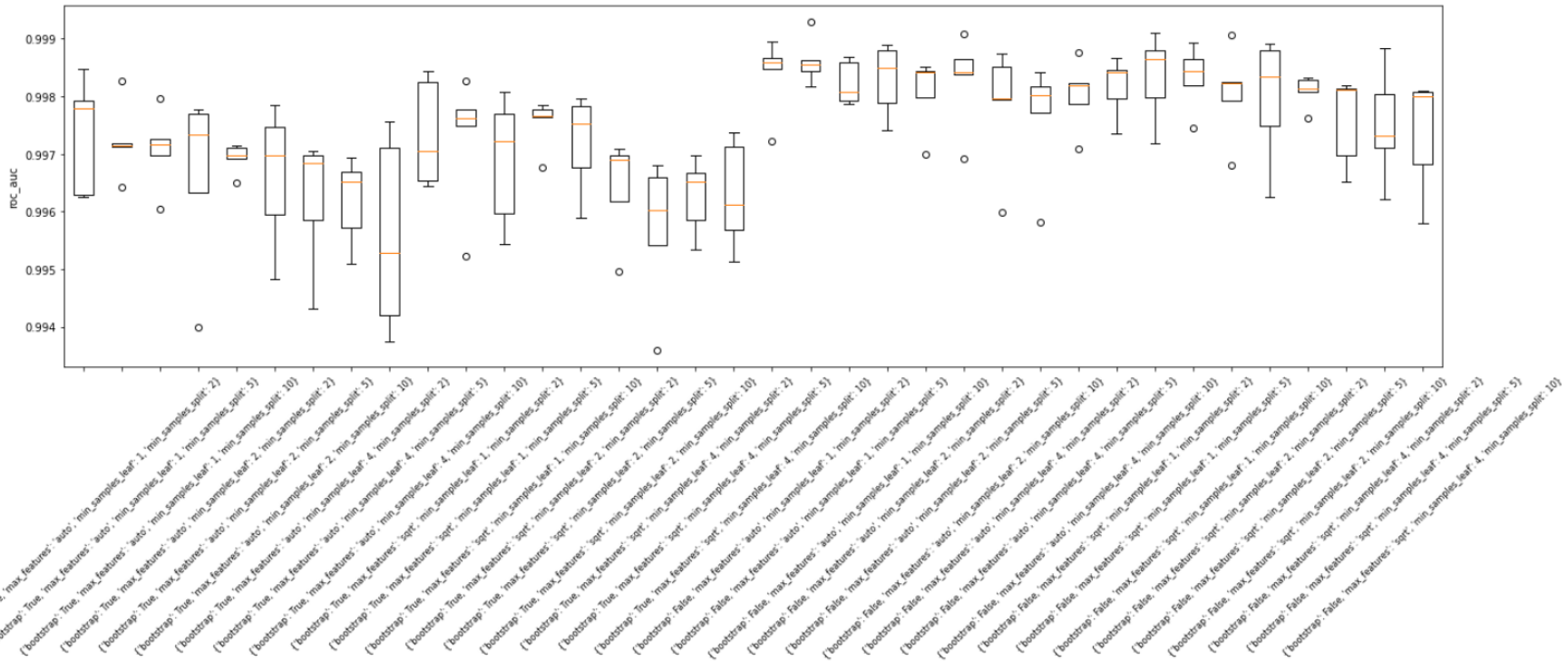
5.2. ANEXO I : ANALISIS DE HIPER PARAMETROS RF


Recall
0.985
0.980
0.975
0.970
0.965
0.960
0.955


Precision
0.985
0.980
0.975
0.970
0.965
0.960
0.955


f1
0.985
0.980
0.975
0.970
0.965
0.960
0.955
{'bootstrap': True, 'max_features': 'auto', 'min_samples_leaf': 1, 'min_samples_split': 2}
{'bootstrap': True, 'max_features': 'auto', 'min_samples_leaf': 1, 'min_samples_split': 5}
{'bootstrap': True, 'max_features': 'auto', 'min_samples_leaf': 1, 'min_samples_split': 10}
{'bootstrap': True, 'max_features': 'auto', 'min_samples_leaf': 2, 'min_samples_split': 2}
{'bootstrap': True, 'max_features': 'auto', 'min_samples_leaf': 2, 'min_samples_split': 5}
{'bootstrap': True, 'max_features': 'auto', 'min_samples_leaf': 2, 'min_samples_split': 10}
{'bootstrap': True, 'max_features': 'auto', 'min_samples_leaf': 4, 'min_samples_split': 2}
{'bootstrap': True, 'max_features': 'auto', 'min_samples_leaf': 4, 'min_samples_split': 5}
{'bootstrap': True, 'max_features': 'auto', 'min_samples_leaf': 4, 'min_samples_split': 10}
{'bootstrap': True, 'max_features': 'sqrt', 'min_samples_leaf': 1, 'min_samples_split': 2}
{'bootstrap': True, 'max_features': 'sqrt', 'min_samples_leaf': 1, 'min_samples_split': 5}
{'bootstrap': True, 'max_features': 'sqrt', 'min_samples_leaf': 1, 'min_samples_split': 10}
{'bootstrap': True, 'max_features': 'sqrt', 'min_samples_leaf': 2, 'min_samples_split': 2}
{'bootstrap': True, 'max_features': 'sqrt', 'min_samples_leaf': 2, 'min_samples_split': 5}
{'bootstrap': True, 'max_features': 'sqrt', 'min_samples_leaf': 2, 'min_samples_split': 10}
{'bootstrap': True, 'max_features': 'sqrt', 'min_samples_leaf': 4, 'min_samples_split': 2}
{'bootstrap': True, 'max_features': 'sqrt', 'min_samples_leaf': 4, 'min_samples_split': 5}
{'bootstrap': True, 'max_features': 'sqrt', 'min_samples_leaf': 4, 'min_samples_split': 10}
{'bootstrap': False, 'max_features': 'auto', 'min_samples_leaf': 1, 'min_samples_split': 2}
{'bootstrap': False, 'max_features': 'auto', 'min_samples_leaf': 1, 'min_samples_split': 5}
{'bootstrap': False, 'max_features': 'auto', 'min_samples_leaf': 1, 'min_samples_split': 10}
{'bootstrap': False, 'max_features': 'auto', 'min_samples_leaf': 2, 'min_samples_split': 2}
{'bootstrap': False, 'max_features': 'auto', 'min_samples_leaf': 2, 'min_samples_split': 5}
{'bootstrap': False, 'max_features': 'auto', 'min_samples_leaf': 2, 'min_samples_split': 10}
{'bootstrap': False, 'max_features': 'auto', 'min_samples_leaf': 4, 'min_samples_split': 2}
{'bootstrap': False, 'max_features': 'auto', 'min_samples_leaf': 4, 'min_samples_split': 5}
{'bootstrap': False, 'max_features': 'auto', 'min_samples_leaf': 4, 'min_samples_split': 10}
{'bootstrap': False, 'max_features': 'sqrt', 'min_samples_leaf': 1, 'min_samples_split': 2}
{'bootstrap': False, 'max_features': 'sqrt', 'min_samples_leaf': 1, 'min_samples_split': 5}
{'bootstrap': False, 'max_features': 'sqrt', 'min_samples_leaf': 1, 'min_samples_split': 10}
{'bootstrap': False, 'max_features': 'sqrt', 'min_samples_leaf': 2, 'min_samples_split': 2}
{'bootstrap': False, 'max_features': 'sqrt', 'min_samples_leaf': 2, 'min_samples_split': 5}
{'bootstrap': False, 'max_features': 'sqrt', 'min_samples_leaf': 2, 'min_samples_split': 10}
{'bootstrap': False, 'max_features': 'sqrt', 'min_samples_leaf': 4, 'min_samples_split': 2}
{'bootstrap': False, 'max_features': 'sqrt', 'min_samples_leaf': 4, 'min_samples_split': 5}
{'bootstrap': False, 'max_features': 'sqrt', 'min_samples_leaf': 4, 'min_samples_split': 10}


accuracy
0.985
0.980
0.975
0.970
0.965
0.960
0.955
{'bootstrap': True, 'max_features': 'auto', 'min_samples_leaf': 1, 'min_samples_split': 2}
{'bootstrap': True, 'max_features': 'auto', 'min_samples_leaf': 1, 'min_samples_split': 5}
{'bootstrap': True, 'max_features': 'auto', 'min_samples_leaf': 1, 'min_samples_split': 10}
{'bootstrap': True, 'max_features': 'auto', 'min_samples_leaf': 2, 'min_samples_split': 2}
{'bootstrap': True, 'max_features': 'auto', 'min_samples_leaf': 2, 'min_samples_split': 5}
{'bootstrap': True, 'max_features': 'auto', 'min_samples_leaf': 2, 'min_samples_split': 10}
{'bootstrap': True, 'max_features': 'auto', 'min_samples_leaf': 4, 'min_samples_split': 2}
{'bootstrap': True, 'max_features': 'auto', 'min_samples_leaf': 4, 'min_samples_split': 5}
{'bootstrap': True, 'max_features': 'auto', 'min_samples_leaf': 4, 'min_samples_split': 10}
{'bootstrap': True, 'max_features': 'sqrt', 'min_samples_leaf': 1, 'min_samples_split': 2}
{'bootstrap': True, 'max_features': 'sqrt', 'min_samples_leaf': 1, 'min_samples_split': 5}
{'bootstrap': True, 'max_features': 'sqrt', 'min_samples_leaf': 1, 'min_samples_split': 10}
{'bootstrap': True, 'max_features': 'sqrt', 'min_samples_leaf': 2, 'min_samples_split': 2}
{'bootstrap': True, 'max_features': 'sqrt', 'min_samples_leaf': 2, 'min_samples_split': 5}
{'bootstrap': True, 'max_features': 'sqrt', 'min_samples_leaf': 2, 'min_samples_split': 10}
{'bootstrap': True, 'max_features': 'sqrt', 'min_samples_leaf': 4, 'min_samples_split': 2}
{'bootstrap': True, 'max_features': 'sqrt', 'min_samples_leaf': 4, 'min_samples_split': 5}
{'bootstrap': True, 'max_features': 'sqrt', 'min_samples_leaf': 4, 'min_samples_split': 10}
{'bootstrap': False, 'max_features': 'auto', 'min_samples_leaf': 1, 'min_samples_split': 2}
{'bootstrap': False, 'max_features': 'auto', 'min_samples_leaf': 1, 'min_samples_split': 5}
{'bootstrap': False, 'max_features': 'auto', 'min_samples_leaf': 1, 'min_samples_split': 10}
{'bootstrap': False, 'max_features': 'auto', 'min_samples_leaf': 2, 'min_samples_split': 2}
{'bootstrap': False, 'max_features': 'auto', 'min_samples_leaf': 2, 'min_samples_split': 5}
{'bootstrap': False, 'max_features': 'auto', 'min_samples_leaf': 2, 'min_samples_split': 10}
{'bootstrap': False, 'max_features': 'auto', 'min_samples_leaf': 4, 'min_samples_split': 2}
{'bootstrap': False, 'max_features': 'auto', 'min_samples_leaf': 4, 'min_samples_split': 5}
{'bootstrap': False, 'max_features': 'auto', 'min_samples_leaf': 4, 'min_samples_split': 10}
{'bootstrap': False, 'max_features': 'sqrt', 'min_samples_leaf': 1, 'min_samples_split': 2}
{'bootstrap': False, 'max_features': 'sqrt', 'min_samples_leaf': 1, 'min_samples_split': 5}
{'bootstrap': False, 'max_features': 'sqrt', 'min_samples_leaf': 1, 'min_samples_split': 10}
{'bootstrap': False, 'max_features': 'sqrt', 'min_samples_leaf': 2, 'min_samples_split': 2}
{'bootstrap': False, 'max_features': 'sqrt', 'min_samples_leaf': 2, 'min_samples_split': 5}
{'bootstrap': False, 'max_features': 'sqrt', 'min_samples_leaf': 2, 'min_samples_split': 10}
{'bootstrap': False, 'max_features': 'sqrt', 'min_samples_leaf': 4, 'min_samples_split': 2}
{'bootstrap': False, 'max_features': 'sqrt', 'min_samples_leaf': 4, 'min_samples_split': 5}
{'bootstrap': False, 'max_features': 'sqrt', 'min_samples_leaf': 4, 'min_samples_split': 10}


roc_auc
0.999
0.998
0.997
0.996
0.995
0.994