UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

Algoritmos de aprendizaje para la clasificación de la retinopatía diabética

TESIS

Que para obtener el título de

Ingeniero en Computación

P R E S E N T A

Brandon Silva Barrera

DIRECTOR DE TESIS

Dr. Ismael Everardo Bárcenas Patiño

Ciudad Universitaria, Cd. Mx., 2024

Investigación realizada gracias al Programa de Apoyo a Proyectos de Investigación e Innovación Tecnológica (PAPIIT) de la UNAM IA104122. Agradezco a la DGAPA-UNAM la beca recibida.

Índice general

Capítulo 1

Resumen

Con este trabajo se busca identificar la capacidad de detectar la retinopatía diabética aplicando Aprendizaje Automatizado a electrorretinogramas (ERG) obtenidos de pacientes sanos y enfermos.

Los ERG recibieron un preprocesamiento a través de la aplicación de la Transformada Continua de Wavelet (TCW) con las wavelet madre Morlet Compleja, Morlet Real, Gaussiana Compleja, Gaussiana Real, Shannon y Sombrero Mexicano. Como resultado de esta operación, se produjo un *escalograma* por cada muestra de ERG, que representa la potencia de una frecuencia respecto al tiempo. De esta representación se obtuvo el promedio de la potencia en 71 frecuencias de interés obtenidas de forma logarítmica. Con esto cada observación o muestra que procesan los modelos tiene 71 variables numéricas y una variable categórica que indica el estado sano o enfermo.

Posteriormente a las observaciones se les aplicaron los algoritmos de extracción de características Análisis de Componentes Principales (ACP) y Análisis Discriminante Lineal (ADL), que son utilizados para comprimir los datos y para hacer más efectivo su uso por parte de los modelos de Aprendizaje Automatizado.

Los algoritmos de Aprendizaje Automatizado aplicados son Bosques Aleatorios, Bayesiano Ingenuo Multinomial, Bayesiano Ingenuo Gaussiano, Redes Neuronales, Redes Neuronales Convolucionales y Redes Neuronales Recurrentes. Para mejorar su desempeño y capacidad de clasificación, a los algoritmos se les aplicó una optimización de hiperparámetros, que es un método que prueba iterativamente distintos valores de las variables iniciales que requiere el funcionamiento interno de un algoritmo, de forma que alguna métrica mejore (estas variables son adicionales a las 71 variables de cada observación y son diferentes para cada algoritmo).

Para analizar cuáles algoritmos y métodos de preprocesamiento de datos fueron más útiles, se hace una comparativa de las métricas abc_cor, exactitud, precisión, sensibilidad, especificidad y f1-score para los modelos con cada una de las 6 wavelet madre utilizadas, con las 71 variables obtenidas con TCW y el promedio de potencias, con ACP y con ADL.

Los resultados muestran que ADL reduce la capacidad de detección y distinción entre sanos

y enfermos. ACP mantiene resultados similares o menos efectivos a comparación de los datos que no lo aplicaron, pero permite comprimir los datos entre el $2\,\%$ y $4\,\%$ del tamaño original. El mejor resultado lo producen las Redes Neuronales con las 71 variables obtenidas de la wavelet Gaussiana Real sin aplicarles ACP o ADL. El peor desempeño lo tienen las Redes Neuronales con la wavelet Gaussiana Compleja aplicando ADL.

Summary

This work seeks to identify the ability to detect diabetic retinopathy by applying Machine Learning to electroretinograms (ERG) obtained from healthy and sick patients.

The ERGs received preprocessing through the application of the Continuous Wavelet Transform (CWT) with the mother wavelets Complex Morlet, Real Morlet, Complex Gaussian, Real Gaussian, Shannon and Mexican Hat. As a result of this operation, a scalogram was produced for each ERG sample, representing the power of a frequency with respect to time. From this representation, the average power was obtained at 71 frequencies of interest obtained logarithmically. With this, each observation or sample processed by the models has 71 numerical variables and a categorical variable that indicates the healthy or sick state.

Subsequently, the feature extraction algorithms Principal Component Analysis (PCA) and Linear Discriminant Analysis (LDA) were applied to the observations, which are used to compress the data and to make its use more effective by the Automated Learning models.

The Machine Learning algorithms applied are Random Forests, Multinomial Naive Bayes, Gaussian Naive Bayes, Neural Networks, Convolutional Neural Networks and Recurrent Neural Networks. To improve their performance and classification capacity, hyperparameter optimization was applied to the algorithms, which is a method that iteratively tests different values of the initial variables required by the internal functioning of an algorithm, so that some metric improves (these variables are additional to the 71 variables of each observation and are different for each algorithm).

To analyze which algorithms and data preprocessing methods were most useful, a comparison is made of the metrics auc_roc, accuracy, precision, sensitivity, specificity and f1-score for the models with each of the 6 mother wavelets used, with the 71 variables obtained with CWT and the average of powers, with PCA and with LDA.

The results show that LDA reduces the ability to detect and distinguish between healthy and sick. PCA maintains similar or less effective results compared to the data that did not apply it, but allows the data to be compressed between $2\,\%$ and $4\,\%$ of the original size. The best result is produced by Neural Networks with the 71 variables obtained from the Real Gaussian wavelet without applying PCA or LDA. The worst performance is achieved by Neural Networks with the Complex Gaussian wavelet applying LDA.

Capítulo 2

Introducción

2.1. Definición del Problema

La diabetes es una patología común en México y el mundo. En 2019 se estimaba que 463 millones de personas padecían diabetes, y se pronostica que para 2045 la cifra se incremente a 700 millones [1]. Esta produce distintos daños, entre ellos la retinopatía diabética, que es una de las principales causas de ceguera ($22.27\,\%$ del total mundial) [2]. Se espera que también se incremente el número de personas diagnosticadas con ella [3] [4].

La retinopatía diabética es producida por daños al sistema vascular ocular y al sistema neurovascular, causando aneurismas, hemorragias, exudados y otros daños en la retina, lo que conlleva una disminución o pérdida de la visión. Se divide en proliferativa (etapa final) y no proliferativa (etapa inicial). La etapa no proliferativa se divide en leve, moderada y severa. Cuando se llega a la etapa proliferativa el daño es irreversible, por lo que es importante hacer un diagnóstico temprano [5] [6] [7].

El Aprendizaje Automatizado consiste en la capacidad de extraer información de un conjunto de datos por medio de algoritmos computacionales con el fin de identificar, optimizar, predecir o clasificar fenómenos [8] [9]. Esta rama de la Inteligencia Artificial ha demostrado tener utilidad en el área médica, aplicándose en el diagnóstico de patologías y su tratamiento [10]. Por medio de esta tecnología se propone desarrollar modelos que permitan detectar en una etapa temprana la retinopatía diabética, permitiendo un tratamiento con mayor antelación.

Para desarrollar los modelos de predicción se tienen como datos iniciales electrorretinogramas obtenidos de pacientes sanos y de pacientes diagnosticados con retinopatía diabética. Un electrorretinograma o electrorretinografía (ERG) de campo completo se define como la medición de la actividad eléctrica en la retina al recibir estímulos de luz, colocando o acercando electrodos en la cornea, la conjuntiva bulbear o la piel del párpado inferior [11]. Las variaciones en la forma e intensidad de las señales eléctricas obtenidas tienen una correlación con la aparición de patologías en la retina y disfunciones en la visión [12]. En capítulos posteriores se explicará con mayor detalle la metodología para recopilar los ERG y sus características.

2.2. Estado del Arte

En el periodo entre 2016 y 2020 la mayoría de las investigaciones de Aprendizaje Automatizado aplicado a la retinopatía diabética se enfocaron al estudio de imágenes de *fondo de ojo* donde se observan daños como aneurismas y hemorragias, procesesándolas en su mayoría con los algoritmos de Árboles de Decisión (AD) y Máquinas de Soporte Vectorial (MSV). Desde 2020 hasta 2022 se dio prioridad a la implementación de Aprendizaje Profundo, Redes Neuronales y métodos de clasificación de imágenes; también sobre fotografías de fondo de ojo [13].

De 150 artículos recopilados sobre la aplicación de Aprendizaje Automatizado a la retinopatía diabética entre enero de 2015 y septiembre de 2020, 81 se enfocaban a la creación de 84 algoritmos para detectar la retinopatía diabética por medio de imágenes de fondo de ojo. Esos algoritmos pertenecen a alguna de las siguientes categorías: Redes Neuronales Convolucionales (RNC), Redes Neuronales Artificiales (RNA), Redes Neuronales Recurrentes (RNR), Redes Neuronales de Memoria a Largo-Corto Plazo (RNMLCP), Bosques Aleatorios (BA), Máquinas de Soporte Vectorial (MSV), métodos de Lógica Difusa y otras configuraciones de Redes Neuronales. Desde 2019 se da un incremento en el uso de RNC. La exactitud o porcentaje de predicciones correctas conseguida para detectar o clasificar la retinopatía diabética con los algoritmos donde se han hecho estudios comparativos está entre $85.96\,\%$ y $99.85\,\%$ [14].

Con la información anterior se puede decir que todavía no se ha hecho un estudio extensivo sobre la posibilidad de utilizar ERG para clasificar la retinopatía diabética a través del Aprendizaje Automatizado. Además las imágenes de fondo de ojo permiten detectar la retinopatía diabética cuando ya existe un daño en el ojo, y un ERG tiene el potencial de hacer una detección antes del daño [15]. A continuación se mencionan algunas investigaciones que han utilizado ERG para clasificar retinopatía diabética o patologías oculares relacionadas.

Se ha diagnosticado glaucoma en ratones a partir de ERG, obteniendo resultados de $90\,\%$ de exactitud para clasificar dos categorías y $80\,\%$ para clasificaciones con más de dos categorías, ambos mediante el algoritmo de Bosques Aleatorios (BA) [16].

Un modelo híbrido que combina los algoritmos de Máquinas de Soporte Vectorial (MSV), Adaboost con Bosques Aleatorios (BA) y Regresión Logística (RL) ha sido utilizado para clasificar fenotipos de ERG según distintas variantes del gen ABCA4, alcanzando una exactitud general de $91.8\,\%$. Para los fenotipos 1, 2 y 3 se obtuvieron $96.7\,\%$, $39.3\,\%$ y $93.8\,\%$ respectivamente [17].

Modelos de Aprendizaje Automatizado fueron aplicados sobre series de tiempo obtenidas de ERG para clasificar neuropatías ópticas. MSV obtuvo una exactitud de $83\,\%$, BA $73\,\%$, Potenciación del Gradiente (PG) $70\,\%$, RNMLCP $68\,\%$, Bosque de Series de Tiempo (BST) $74\,\%$, Vecino Más Cercano (VMC) combinado con Deformación Dinámica del Tiempo (DNT) $65\,\%$ y MSV con Kernel de Función de Base Radial (KFBR) $69\,\%$ [18],

Ha sido comprobado que al aplicar una Transformada Continua de Wavelet (TCW) sobre las señales ERG, la capacidad para detectar una patología se incrementa en el caso del algoritmo de

Árboles de Decisión (AD). La exactitud para clasificar entre pacientes sanos y con una patología está entre $78\,\%$ y $83\,\%$ para un grupo de adultos y entre $58\,\%$ y $70\,\%$ para un grupo pediátrico. Sin la TCW se tiene un $52\,\%$ para el conjunto de datos de ERG pediátrico y $54\,\%$ para el de adultos [19].

Imágenes de fondo de ojo se han combinado con ERG para clasificar la etapa de retinopatía diabética en la que se encuentra un paciente. Por medio de una Red Neuronal Convolucional Profunda (RNCP) se clasifican las imágenes y con una Red Neuronal Artificial (RNA) se clasifican señales ERG. Después se comparan los resultados de ambos modelos para hacer una clasificación final. La exactitud tiene valores cercanos a $96\,\%$ [20].

La diabetes tipo 2 fue diagnosticada a partir de ERG con el algoritmo de Bosques Aleatorios (BA). Para señales obtenidas de ratones la exactitud tiene valores entre $80\,\%$ y $87.5\,\%$; Con señales provenientes de humanos $67.4\,\%$ para datos obtenidos de diferentes clínicas y $86.6\,\%$ con señales medidas desde un mismo dispositivo [21].

Las ventajas clínicas de utilizar ERG a comparación de otros métodos son el ser un proceso no invasivo para el paciente que puede ser realizado en unos minutos y no inhabilita o daña al paciente. El procedimiento para medir un ERG es sencillo y es posible que personal no tan especializado lo aplique. Un ERG puede detectar cambios sutiles en la retina [22]. Por otro lado, los dispositivos de obtención de ERG más modernos son portátiles y no requieren colocar electrodos directamente en los ojos del paciente.

Como aportación a la investigación del Aprendizaje Automatizado con ERG para clasificar retinopatía diabética, se propone evaluar el desempeño de distintos algoritmos de clasificación que utilicen como entrada ERG a los que se les han extraído características relevantes a través de algoritmos como el Análisis de Componentes Principales y el Análisis Discriminante Lineal.

2.3. Objetivos

- Desarrollar modelos de Aprendizaje Automatizado para detectar la retinopatía diabética a partir de ERG.
    - Implementar los algoritmos Bosques Aleatorios, Bayes Ingenuo Multinomial, Bayes Ingenuo Gaussiano, Redes Neuronales simples, Redes Neuronales Recurrentes y Redes Neuronales Convolucionales para la detección de retinopatía diabética a través de ERG.
    - Optimizar los algoritmos de Bosques Aleatorios, Bayes Ingenuo Multinomial, Bayes Ingenuo Gaussiano, Redes Neuronales simples, Redes Neuronales Recurrentes y Redes Neuronales Convolucionales a través de extracción de características y optimización de hiperparámetros.

- Comparar el desempeño de los modelos al preprocesar los ERG que reciben con la Transformada Continua de Wavelet.
    - Utilizar ERG preprocesados con las wavelet madre Morlet Compleja, Morlet Real, Gaussiana Compleja, Gaussiana Real, Shannon y Sombrero Mexicano.
    - Obtener métricas que describan el desempeño de los modelos con cada una de las wavelet madre aplicadas, con el fin de conocer cuáles wavelet madre tienen mayor potencial para facilitar la detección de retinopatía diabética.

2.4. Contenido de esta obra

En el capítulo 1 y 2 se da un resumen y una introducción general acerca del estudio realizado en este texto.

En el capítulo 3 se muestra la teoría y conceptos que son el fundamento para la investigación mostrada en esta obra, que son la definición y descripción del Aprendizaje Automatizado, los algoritmos de aprendizaje utilizados y su funcionamiento, las técnicas para manejar o filtrar los datos en un estudio como el aquí realizado, con un enfoque en el Análisis de Componentes Principales (ACP) y el Análisis Discriminante Lineal (ADL), además de las métricas para evaluar un modelo de Aprendizaje Automatizado, el método de optimización de hiperparámetros que permite mejorar las métricas de un algoritmo de aprendizaje y la Transformada Continua de Wavelet (TCW), que permitió filtrar y preparar los electrorretinogramas (ERG) utilizados en este estudio.

El origen, método de obtención y preprocesamiento de los ERG es explicado en el capítulo 4. En el capítulo 5 se da una descripción más detallada de cómo se implementaron los algoritmos de Aprendizaje Automatizado, los métodos de extracción de características ACP y ADL, y la optimización de hiperparámetros.

El capítulo 6 contiene la descripción y análisis de los resultados obtenidos en los experimentos realizados para este trabajo, apoyándose de gráficas de calor que permiten comparar el desempeño de determinado algoritmo con un método de extracción de características y cada wavelet. Cada resultado es un promedio de 10 repeticiones de algún algoritmo.

El capítulo 7 contiene las conclusiones de esta investigación. Finalmente, en el apéndice A se colocan las métricas obtenidas en cada experimento realizado, de los cuales surgieron los promedios. El apéndice B contiene tablas con los valores de hiperparámetros obtenidos para cada algoritmo y el apéndice C gráficas que representan las mejoras obtenidas con la optimización de hiperparámetros.

Capítulo 3

Marco Teórico

3.1. Definición, clasificación y características del Aprendizaje Automatizado

El Aprendizaje Automatizado surge como un subcampo de la Inteligencia Artificial. Este consiste en determinar o predecir una característica, comportamiento o pertenencia con algoritmos o modelos matemáticos, según una o distintas *variables objetivo* (variables dependientes) que obtienen sus valores a partir de *variables predictoras* (variables independientes) que se originan al procesar un conjunto de datos [23].

Cada proyecto de Aprendizaje Automatizado es único, ya que al trabajar con datos diferentes se producen modelos diferentes [24]. Se divide en Aprendizaje Supervisado, Aprendizaje no Supervisado y Aprendizaje de Refuerzo [25] [26] [27].

3.1.1. Aprendizaje Supervisado

Para este tipo de aprendizaje los algoritmos se aplican sobre *observaciones*, donde cada observación representa un conjunto de variables predictoras y variables objetivo. Cada observación tiene las mismas variables objetivo y variables predictoras que las demás. Las observaciones con las que se formará o *entrenará* un modelo están *etiquetadas*, lo que quiere decir que debe conocerse el valor de las variables objetivo desde un principio. Existen dos tipos de aprendizaje supervisado, clasificación y regresión [28] [29].

Clasificación

Las variables objetivo que se van a predecir son *categóricas*, lo que implica que son finitas, discretas y sin una relación de orden. Las variables predictoras pueden tener cualquier valor que pueda ser procesado o interpretado por un algoritmo.

Regresión

En este caso las variables objetivo se definen como valores numéricos continuos dependientes y las variables predictoras como valores numéricos continuos independientes. Estas variables tienen una relación matemática que se establece o encuentra mediante los modelos de Aprendizaje Supervisado y se representa con una función o modelo matemático [25].

3.1.2. Aprendizaje no Supervisado

Los datos u observaciones usados en este tipo de aprendizaje no tienen etiquetado, ya que no se conoce la clasificación o estructura de los datos. Mediante este método se trata de encontrar un patrón o característica de los datos que permita usarlos para clasificar o que haga posible etiquetarlos [29]. Un algoritmo perteneciente a este tipo de aprendizaje es el *agrupamiento*, en el cual se forman grupos de datos que comparten valores o alguna característica. También se dice que los algoritmos para reducir la dimensionalidad de los datos se pueden considerar parte del aprendizaje no supervisado [30].

Algunos autores agregan la clasificación del Aprendizaje Semi-supervisado, donde para un conjunto de observaciones sólo se tiene una parte etiquetada. Las observaciones no etiquetadas se agrupan de manera no supervisada comparándose con los datos ya etiquetados, de manera que por cada valor o valores posibles de las variables objetivo hay un grupo de observaciones no etiquetadas. Después se etiquetan las observaciones según el grupo al que pertenezcan, hasta que todas las observaciones existentes estén etiquetadas y puedan ser usadas en el aprendizaje supervisado [31].

3.1.3. Aprendizaje por Refuerzo

Consiste en un sistema o *agente* que se mejora a sí mismo después de repetir determinadas acciones en su entorno, las cuales le devuelven valores que le indican si sus acciones fueron las correctas o esperadas. El entorno se divide en *estados* a los que se llega mediante determinadas acciones, y es posible establecer un estado final que se alcanza o evita. Estos modelos se entrenan a través de varias iteraciones en las que se encuentran las mejores acciones para llegar al estado final o deseado, perfeccionándose a sí mismos en cada iteración [32] [33].

3.1.4. Etapas del Aprendizaje Automatizado

Los autores no concuerdan con la definición y cantidad de etapas [34] [35], sin embargo, se pueden estructurar como las siguientes según las fuentes de información consultadas:

Extracción o selección de características. Variables a utilizar

Generalmente los datos o muestras disponibles para un estudio están mezclados con datos irrelevantes o no son los óptimos para un objetivo de investigación establecido, por lo que

es necesario realizar un análisis que permita conocer los datos existentes y a partir de ellos seleccionar los más adecuados. Este análisis se conoce como extracción de características, y si no se hace de manera correcta, el desempeño de los algoritmos de aprendizaje será afectado [36].

También es importante evitar *sesgos cognitivos* al conformar un conjunto de datos. Estos sesgos se producen debido a inferencias o creencias de las personas que hacen que discriminen o favorezcan ciertos datos. Si los datos no se eligen imparcialmente, las predicciones y resultados de los modelos no mostrarán efectivamente la realidad del fenómeno que se esté estudiando [37].

Se han desarrollado técnicas que potencian la extracción de características según el tipo de datos que se manejen, como las orientadas a las series de tiempo, las enfocadas a señales de frecuencia o las diseñadas para tratar imágenes [36]. Conocer profundamente los datos disponibles permitirá elegir las técnicas más adecuadas, lo que incrementará la posibilidad de tener buenos resultados de predicción o clasificación.

En algunas fuentes se hace una diferencia entre los conceptos de *selección de características* y *extracción de características*. El primero se describe como escoger algunas variables predictoras según un criterio como la relevancia o correlación respecto de las demás, y el segundo como la aplicación de un algoritmo que obtenga nuevas variables predictoras a partir de procesar las originales. Cualquiera de los dos métodos mencionados busca reducir la dimensionalidad de los datos para hacer más precisos o rápidos los modelos de Aprendizaje Automatizado [38] [39].

Selección de algoritmo de aprendizaje

Se dice que no hay una técnica precisa para escoger el algoritmo de aprendizaje que mejor funcionará en un estudio, por lo que se deben probar varios algoritmos y obtener su desempeño según una métrica como la exactitud [40] [41]. Si un algoritmo resultó ser el mejor en una investigación, no necesariamente es el mejor para otras investigaciones similares. Cada algoritmo tiene sus ventajas y desventajas debido a la forma en que funciona. Por otra parte, el tipo o formato de los datos que recibe puede afectar su rendimiento [42].

Preprocesamiento

Si los datos no tienen un formato de valor útil para un algoritmo de aprendizaje automatizado, después de recopilarlos y organizarlos habrá que hacer transformaciones sobre ellos con el fin de hacerlos aptos. Algunas de estas transformaciones son la estandarización según una distribución normal o el escalado de los datos en un intervalo de valores de 0 a 1. También se recomienda aplicar algoritmos de reducción de dimensionalidad, que disminuirán la posibilidad de tener datos redundantes o correlacionados y agilizarán la ejecución de los algoritmos de aprendizaje al compactar el tamaño de los datos que procesarán [43].

Algunos autores consideran que esta etapa es perteneciente o igual a la extracción de características [38].

Aprendizaje o entrenamiento

Cuando a partir de un conjunto de observaciones se crea un modelo matemático que permite predecir o clasificar las variables objetivo de otro conjunto de observaciones, se dice que el modelo *aprende* o se *entrena*. Para evitar sesgos estadísticos o sobreajustes (cuando un modelo clasifica o predice muy bien los datos con los que fue entrenado pero no otros) al evaluar el desempeño de un modelo o algoritmo de aprendizaje supervisado, se dividen los datos preprocesados en dos conjuntos, uno de *entrenamiento* y otro de *prueba*.

Si no se usan conjuntos de entrenamiento y prueba no se podrá hacer un análisis del desempeño general del modelo, en caso de no tener acceso a nuevas observaciones con las cuales probar un modelo. Las proporciones más utilizadas para los datos de entrenamiento y prueba son $60\,\%/40\,\%$, $70\,\%/30\,\%$ y $80\,\%/20\,\%$ respectivamente. Para conjuntos de datos con más de 100000 observaciones es factible usar la proporción $90\,\%/10\,\%$ ó $99\,\%/1\,\%$ [44].

Con las observaciones del conjunto de entrenamiento se forma un modelo matemático que permitirá predecir o clasificar las observaciones del conjunto de prueba. Esto permite aplicar una validación cruzada donde después de formar un modelo con los datos de entrenamiento, se verifica si logra predecir correctamente las variables objetivo de los datos del conjunto de prueba [40].

Evaluación

El desempeño de un algoritmo de Aprendizaje Automatizado suele medirse al comparar los valores reales de las variables objetivo de cada observación con el valor predecido por el algoritmo [45]. Además de la exactitud con la que un modelo identifica correctamente las variables objetivo de una muestra, un modelo puede evaluarse según los recursos computacionales que demanda o qué tan bien se puede interpretar la información que devuelve [42]. Algunas veces se necesita predecir en tiempo real, así que se ajusta un modelo con el fin de equilibrar la exactitud de predicción y el tiempo de procesamiento requerido para producirla [37].

Existen distintas métricas de evaluación para conocer el desempeño de los resultados de la predicción de un modelo. Más adelante se explicarán las métricas utilizadas en este estudio.

Predicción o aprovechamiento de resultados

La exactitud o el porcentaje de acierto al predecir un elemento en una muestra es variable, por lo que se necesita establecer reglas para decidir qué hacer con el resultado de la predicción. Por ejemplo, si se determina que un elemento es $80\,\%$ A y $20\,\%$ B, tomarlo como A.

3.2. Algoritmos

3.2.1. Bosques Aleatorios

Los bosques aleatorios [46] se componen de varios árboles de decisión. Un árbol de decisión inicialmente divide un grupo de datos u observaciones de entrenamiento en dos conjuntos siguiendo alguna regla básica como $<=$ y $>=$ o algunas otras más elaboradas. Posteriormente a cada conjunto lo vuelve a dividir en dos subconjuntos, y así sucesivamente, hasta que según una condición no sea necesario producir más subconjuntos.

Un árbol de decisión que compone a un bosque aleatorio está construido con una muestra aleatoria del total de los datos de entrenamiento, recopilada según el método *bootstrap* (Una observación puede aparecer varias veces en la muestra al extraerse con reemplazo). Además la muestra utilizada para cada árbol debe de tener la misma distribución de probabilidad que las demás y cada nodo de los árboles es un subconjunto aleatorio de variables predictoras.

Cada árbol de decisión emite un *voto* en el que se dice a qué valor de variable objetivo pertenece su subconjunto de muestras y variables predictoras. Para predecir las variables objetivo de las observaciones de prueba se toman todos los votos de los árboles y por medio de una validación cruzada se decide su valor.

Los bosques aleatorios pueden utilizarse para clasificación o regresión [47] [48]. Para observaciones con variables objetivo categóricas se decide según el valor que tenga más votos y para observaciones con variables objetivo numéricas se decide con un promedio de los votos.

3.2.2. Bayesiano Ingenuo

Surge de la teoría bayesiana aplicada a las observaciones en los datos [49]. Tomando una observación como una tupla $O = (x_1, x_2, ..., x_n)$ donde cada x_n es una variable predictora, y también a una variable objetivo C con un valor c, se obtiene que la probabilidad de que una observación O tenga como valor de variable objetivo a c es la siguiente:

$$P(c|O) = \frac{P(O|c)P(c)}{P(O)} \tag{3.1}$$

Si el valor de C se considera verdadero o falso, es sólo verdadero si se cumple lo siguiente:

$$f_b(O) = \frac{P(C = verdadera|O)}{P(C = falsa|O)} >= 1$$

Siendo $f_b(O)$ un clasificador bayesiano.

Si para la implementación del clasificador se establece que las variables predictoras son independientes probabilísticamente, dada la variable objetivo c, se obtiene un clasificador bayesiano ingenuo y se puede considerar lo siguiente:

$$P(O|c) = P(x_1, x_2, ..., x_n|c) = \prod_{i=1}^{n} p(x_i|c)$$

Lo que implica que el clasificador bayesiano ingenuo se expresa así:

$$f_b(O) = \frac{P(C = verdadera|O)}{P(C = falsa|O)} \prod_{i=1}^{n} \frac{p(x_i|C = verdadera)}{p(x_i|C = falsa)}$$

Bayesiano Ingenuo Multinomial

Toma en cuenta el total de apariciones de algún valor en las variables predictoras y cuántas de esas apariciones pertenecen a determinada variable objetivo [50].

$$P(x_{i-valor_j}|c) = \frac{Contar(x_{i-valor_j}, c) + 1}{Contar(c) + |V|}$$

Donde $Contar(x_{i-valor_j}, c)$ es el total de veces que algún valor j de las variables predictoras de todas las observaciones pertenece a la categoría c, $Contar(c)$ es la cantidad de valores diferentes que aparecen en las variables predictoras de las observaciones que se etiquetan como c y $|V|$ es el número de valores diferentes que tienen las variables predictoras de todas las observaciones.

Para clasificar una observación O se utiliza la siguiente ecuación para cada valor c que puedan tomar las variables objetivo:

$$P(c_k|O) \propto P(c_k) \prod_{i=1}^{n} P(x_{i-valor_j}|c_k)$$

Asignándose el valor c que tenga la mayor probabilidad:

$$argmax(P(c_1|O), P(c_2|O), ..., P(c_k|O))$$

Bayesiano Ingenuo Gaussiano

Considera una distribución de probabilidad gaussiana para las variables predictoras:

$$P(x_i|C) = \frac{1}{\sqrt{2\pi\sigma_C^2}} e^{(-\frac{(x_i-\mu_C)^2}{2\sigma_C^2})}$$

donde σ_C^2 es la desviación estándar y μ_C es la media estándar.

3.2.3. Redes Neuronales

Las redes neuronales surgen del concepto de neurona artificial o perceptron, que se define como una combinación lineal de entradas x con un peso correspondiente w [51][52].

$$w = \begin{bmatrix} w_1 \\ \vdots \\ w_m \end{bmatrix}, \qquad x = \begin{bmatrix} x_1 \\ \vdots \\ x_m \end{bmatrix}$$

La combinación lineal es llamada la entrada neta: $z = w_1x_1 + w_2x_2 + ... + w_mx_m$. Esta entrada se pasa a una función de decisión $\phi(z)$ que activa o desactiva la neurona según un límite θ. Por ejemplo:

$$\phi(z) = \begin{cases} 1 & si \quad z \geq \theta \\ \\ -1 & si \quad z < \theta \end{cases}$$

Por convención el límite θ se introduce en la ecuación de z como un peso $w_0 = -\theta$, multiplicado por una entrada $x_0 = 1$, lo que simplifica a la ecuación de z como:

$$z = w_0x_0 + w_1x_1 + ... + w_mx_m = \mathbf{W^t x}.$$

Lo que permite expresar a $\phi(z)$ como:

$$\phi(z) = \begin{cases} 1 & si \quad z \geq 0 \\ \\ -1 & si \quad z < 0 \end{cases}$$

El peso $w_0 = -\theta$ se conoce como la *unidad de sesgo*.

El perceptron definido permite agrupar las entradas que recibe en dos conjuntos, las mayores a 0 y las menores a 0. El concepto de perceptron se puede especializar para lograr un aprendizaje y realizar clasificaciones más complejas al entrenarlo con observaciones etiquetadas.

Si los pesos w se inicializan a 0 y una observación de algún banco de datos se toma como la entrada x, se es capaz de utilizar el valor de activación $\phi(z)$ como una salida $\hat{y}$ que actualiza los pesos w para multiplicar una nueva observación x. La actualización se define como:

$$w_j := w_j + \Delta w_j, \qquad \Delta w_j = \eta(y^{(i)} - \hat{y}^{(i)})x_j^{(i)}$$

Donde Δw_j es el peso w actualizado, η es la tasa de aprendizaje (constante entre 0.0 y 1.0), $y^{(i)}$ es el valor verdadero de la variable objetivo de la observación y $\hat{y}^{(i)}$ es el valor de activación $\phi(z)$, que se toma como el valor de variable objetivo predecido.

Para este ejemplo de perceptron donde se clasifican las observaciones en dos categorías, sólo se podrán clasificar si las observaciones son linealmente separables, ya que cada perceptrón es equiparable a una función linealmente separable. Las salidas de los perceptrones son linealmente separables ya que pueden colocarse en una gráfica de dos dimensiones y separarse con una línea recta, de manera que de un lado queda el valor -1 y en otro 1 [53]. A la capacidad de clasificar totalmente las observaciones se le conoce como *convergencia*.

Cuando se colocan varios perceptrones unos tras otros en distintas capas se tiene una red neuronal. La primera capa se conoce como la capa de entrada y la última como capa de salida; Las capas intermedias son nombradas *capas escondidas.* Si hay más de una capa escondida se tiene una *Red Neuronal Artificial Profunda* (RNAP) (Figura 3.1).

Como las salidas de los perceptrones (llamados también *unidades*) pasan desde la capa de entrada a las capas escondidas y finalmente a la capa de salida, se dice también que es una *Red Neuronal Alimentada Hacia Adelante* (RNAHD).

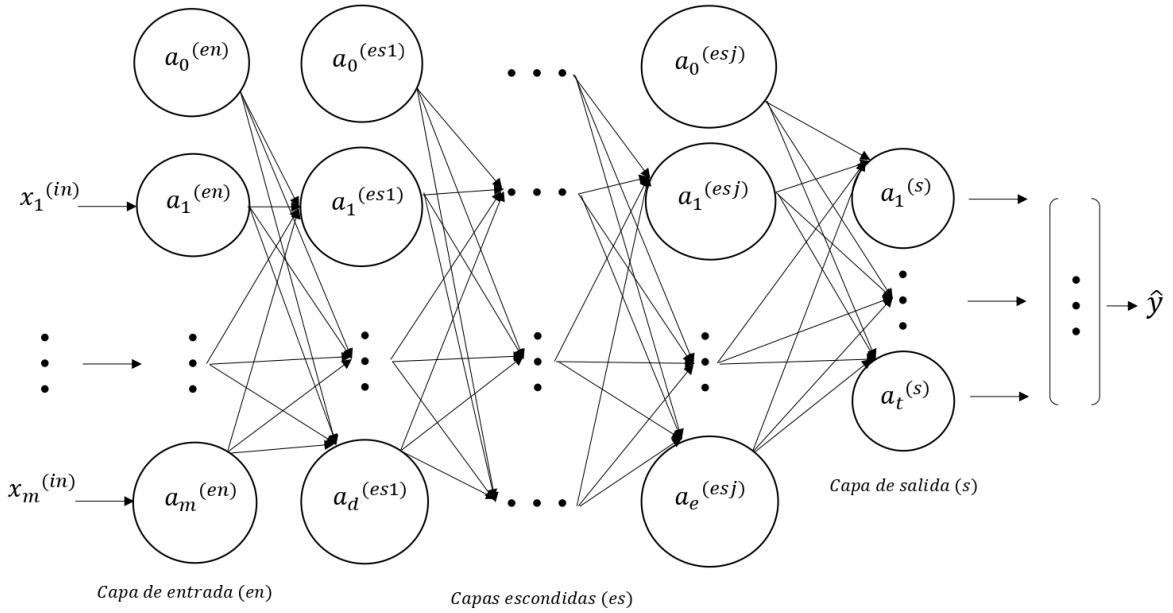


Figura 3.1: Red Neuronal Artificial Profunda

Algoritmo de Propagación Hacia Atrás

Cuando una RNAP devuelve una o varias salidas (una por cada variable objetivo), lo cual se expresa como: $\hat{y} = (y_1, y_2, ..., y_s)$, se puede calcular un error $e_s = voo_s - y_s$, donde voo_s es la s-ésima variable objetivo de una observación. Lo anterior permite comparar los valores verdaderos de las variables objetivo de una observación con los valores de variables objetivo que se aproximan con la salida $\hat{y}$. El error general o total de la RNAP para una observación se puede definir como:

$$E = \frac{1}{2}\sum_{i=1}^{s} e_i^2$$

Después, para actualizar cada uno de los pesos y unidades de sesgo de la red neuronal $W_{[u_u,v_u]}$ donde $[u_u, v_u]$ es el conjunto de conexiones entre las neuronas, se utiliza la siguiente ecuación, que representa la relación entre los pesos de la red neuronal y el error provocado:

$$\Delta W_{[u_u,v_u]} = -\eta \frac{\partial E}{\partial W_{[u_u,v_u]}}$$

Esta ecuación surge de aplicar la *regla de la cadena* sobre las derivadas parciales de los pesos y unidades de sesgo de los perceptrones anteriores a un perceptron al que se le está actualizando el peso. El resultado de esta multiplicación de derivadas parciales se ajusta con la tasa de aprendizaje η.

Los nuevos valores de $W_{[u_u,v_u]}$ se obtienen con:

$$W_{[u_u,v_u]nuevos} = W_{[u_u,v_u]} + \Delta W_{[u_u,v_u]}$$

Al tener los nuevos pesos se da otra observación a la RNAP y se obtiene el nuevo valor E, con el cual se pueden volver a calcular los pesos. Este proceso se detiene hasta que el error

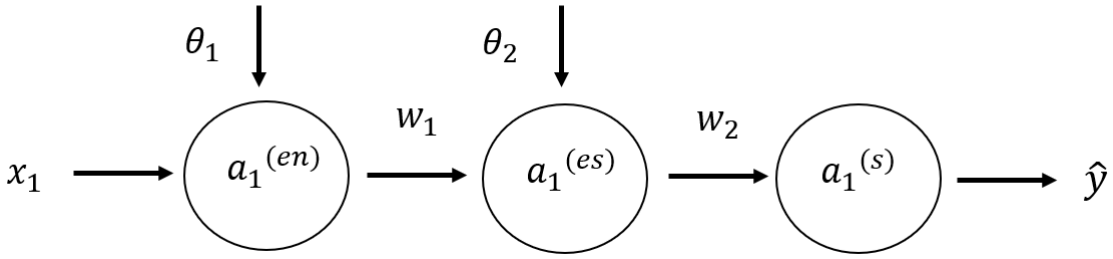


Figura 3.2: Red Neuronal simple de tres capas con un sólo perceptron por cada una.

no sobrepase un valor máximo predefinido, manteniendo los últimos pesos calculados para predecir las variables objetivo de nuevas observaciones [54]. A la acción de actualizar los pesos cada que se procesa una sola observación elegida aleatoriamente se le conoce como *Gradiente Descendente Estocástico*, y es el método de Propagación Hacia Atrás más utilizado y aceptado [55] [56]. Cuando todas las observaciones de entrenamiento han sido utilizadas para actualizar los pesos se dice que se ha cumplido una *época*, ya que al final de esta se compara el error general de la predicción de la última observación con un valor de *tolerancia*. Si el error general sigue siendo mayor a la tolerancia, se ejecuta otra época, repitiendo el ajuste de los pesos.

Además existen los métodos de *Lote* o *Determinante Estocástico* y *Minilote*, donde en el primero se actualizan los pesos operando todas las observaciones al mismo tiempo y en el segundo se actualizan con varias al mismo tiempo [57].

A continuación se desarrollará un ejemplo del algoritmo de Propagación Hacia Atrás con el método del Gradiente Descendente Estocástico para una Red Neuronal simple con una capa de entrada, una capa escondida con función de decisión o activación y una capa de salida (Figura 3.2).

Se parte de una observación X con una sola variable predictora x_1 con valor de 5 y una sola variable objetivo y con valor de 10, para unos pesos iniciales y unidades de sesgo con valores inciales aleatorios $w_1 = 0.5$, w_2 = 0.8, $\theta_1 = 0.7$ y $\theta_2 = 0.2$. El resultado de la primera capa es:

$$a_1{}^{(en)} = (5 \cdot 0.5) + 0.7 = 3.2$$

Para la segunda:

$$a_1{}^{(es)} = (3.2 \cdot 0.8) + 0.2 = 2.76$$

Con lo que la capa de salida devuelve el valor de:

$$\hat{y} = 2.76$$

Para actualizar w_2 se obtiene la derivada parcial del error general respecto a w_2:

$$\frac{\partial E}{\partial w_2} = \frac{\partial E}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial w_2}$$

E se simplifica al tener sólo un valor de variable objetivo:

$$E = \frac{1}{2} \sum_{i=1}^{s} e_i^2 = \frac{1}{2} (\hat{y} - y)^2$$

Con lo que la derivada parcial respecto a $\hat{y}$ queda como:

$$\frac{\partial E}{\partial \hat{y}} = (\hat{y} - y)$$

$\hat{y}$ es igual a:

$$\hat{y} = w_2 \cdot {a_1}^{(en)} + \theta_2$$

Así que su derivada parcial respecto a w_2 es:

$$\frac{\partial \hat{y}}{\partial w_2} = {a_1}^{(en)}$$

Con lo que:

$$\frac{\partial E}{\partial w_2} = \frac{\partial E}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial w_2} = (\hat{y} - y) \cdot {a_1}^{(en)} = (2.76 - 10) \cdot 2.76 = -19.9824$$

Para actualizar θ_2 se sigue el mismo procedimiento y se obtiene que:

$$\frac{\partial E}{\partial \theta_2} = \frac{\partial E}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial \theta_2} = (\hat{y} - y) \cdot 1 = (2.76 - 10) = -7.24$$

Suponiendo una tasa de aprendizaje η de 0.001, los nuevos valores de w_2 y θ_2 son:

$$w_{2 \quad nuevo} = 0.8 - (0.001 \cdot -19.9824) = 0.82$$

$$\theta_{2 \quad nuevo} = 0.2 - (0.001 \cdot -7.24) = 0.2072$$

Ahora para calcular los nuevos w_1 y θ_1 la regla de la cadena debe extenderse hacia la capa anterior (de ahí el nombre de Propagación Hacia Atrás):

$$\frac{\partial E}{\partial w_1} = \frac{\partial E}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial {a_1}^{(en)}} \cdot \frac{\partial {a_1}^{(en)}}{\partial w_1}$$

$$= (\hat{y} - y) \cdot {a_1}^{(en)} = (2.76 - 10) \cdot 2.76 = -19.9824$$

Dado que la capa escondida ${a_1}^{(en)}$ tiene una función de activación $\phi(CL)$, la derivada parcial que le corresponde debe dividirse en dos, con una variable CL que representa el valor devuelto por la combinación lineal del perceptron que debe introducirse en la función de activación:

$$\frac{\partial E}{\partial w_1} = \frac{\partial E}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial {a_1}^{(en)}} \cdot \frac{\partial {a_1}^{(en)}}{\partial CL} \cdot \frac{\partial CL}{\partial w_1}$$

Dado que $\hat{y} = w_2 \cdot {a_1}^{(en)} + \theta_2$, se tiene el siguiente resultado:

$$\frac{\partial \hat{y}}{\partial {a_1}^{(en)}} = w_2$$

Estableciendo la función de activación como la función identidad $\phi(CL) = CL$ y tomando en cuenta que ${a_1}^{(en)} = \phi(CL)$:

$$\frac{\partial {a_1}^{(en)}}{\partial CL} = \frac{\partial \phi(CL)}{\partial CL} = 1$$

Como $CL = (x_1 \cdot w_1) + \theta_1$, se puede establecer que:

$$\frac{\partial CL}{\partial w_1} = x_1$$

Así que:

$$\frac{\partial E}{\partial w_1} = (\hat{y} - y) \cdot w_2 \cdot 1 \cdot x_1 = (2.76 - 10) \cdot 0.8 \cdot 5 = -28.96$$

En el caso de θ_1 se aplica el mismo procedimiento:

$$\frac{\partial E}{\partial \theta_1} = \frac{\partial E}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial {a_1}^{(en)}} \cdot \frac{\partial {a_1}^{(en)}}{\partial \theta_1}$$

$$\frac{\partial E}{\partial \theta_1} = \frac{\partial E}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial {a_1}^{(en)}} \cdot \frac{\partial {a_1}^{(en)}}{\partial CL} \cdot \frac{\partial CL}{\partial \theta_1}$$

Con lo que se obtiene:

$$\frac{\partial E}{\partial \theta_1} = (\hat{y} - y) \cdot w_2 \cdot 1 \cdot 1 = (2.76 - 10) \cdot 0.8 = -5.792$$

Actualizando w_1 y θ_1

$$w_1 \quad _{nuevo} = 0.5 - (0.001 \cdot -28.96) = 0.5290$$

$$\theta_1 \quad _{nuevo} = 0.7 - (0.001 \cdot -5.792) = 0.7058$$

Después se repite todo el procedimiento o se ejecuta otra época con una observación aleatoria diferente a la actual, y así consecutivamente hasta que se consiga un error general menor a alguna tolerancia fijada.

3.2.4. Redes Neuronales Recurrentes (RNR)

Este tipo de redes neuronales están diseñadas para trabajar con información secuencial, como series de tiempo y estructuras de datos que tienen valores que están relacionados con los valores que estén antes y después de ellos, formando un significado en conjunto, como audio, texto y video. Las RNR se diferencian de las RNAP en que la primera considera una dependencia o relación entre las características secuenciales y temporales de los datos de entrada y la segunda considera a las entradas independientes entre ellas [58]. Otra diferencia es que en las RNR la salida de una capa escondida no solamente puede enviarse a la capas siguientes, sino también a las anteriores o a sí misma.

Las RNR pueden explicarse como una RNAP que actualiza las características o valores de los perceptrones de sus capas escondidas según el resultado de las características iniciales (Figura 3.3). Matemáticamente se expresa como lo siguiente [59][60]:

$$A^{(t)} = \phi(W_A A^{(t-1)} + W_X X^{(t)} + \theta_A)$$

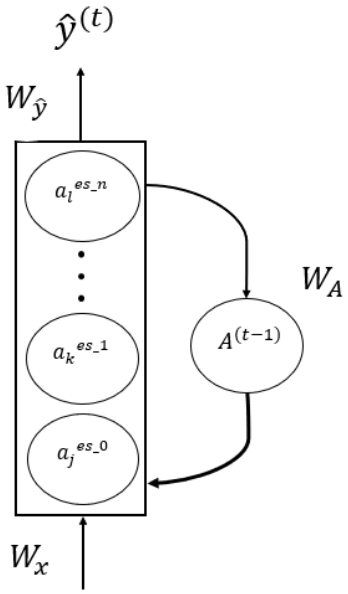


Figura 3.3: Red Neuronal Recurrente

$$\hat{y}^{(t)} = \phi(W_{\hat{y}} A^{(t)} + \theta_{\hat{y}})$$

En donde t es un estado o configuración de las capas escondidas, $t-1$ es un estado anterior, $A^{(t)}$ es el resultado devuelto por las capas escondidas antes de introducirlo a la capa de salida, $X^{(t)}$ son las observaciones procesadas por la red, $\hat{y}^{(t)}$ es la salida o la predicción hecha, W_X es un vector con los pesos en la capa de entrada, $W_{\hat{y}}$ contiene los pesos en la capa de salida, W_A los pesos de las capas escondidas de un estado anterior al actual, ϕ es una función de activación, θ_A es la unidad de sesgo de las capas escondidas y $\theta_{\hat{y}}$ es la unidad de sesgo de la capa de salida.

En el primer estado de la red los valores de $A^{(t)}$ se inicializan a 0. Además se potencia el funcionamiento de una RNR actualizando independientemente los tres conjuntos de pesos. Al obtener la salida $\hat{y}$ en un estado, se aplica el Algoritmo de Propagación Hacia Atrás para cada vector de pesos antes de pasar al siguiente estado.

3.2.5. Redes Neuronales Convolucionales (RNC)

Este tipo de redes funciona bien para datos que están organizados en cuadrículas, series o arreglos multidimensionales, donde un valor forma un significado en conjunto con otros dependiendo su posición respecto a otros valores. Un ejemplo de estos datos son las imágenes formadas por pixeles, donde cada pixel por sí solo puede no tener un significado, pero en conjunto describen una figura u objeto. A los arreglos multidimensionales se les llama *tensores* [61].

La característica principal de este tipo de redes es la aplicación de la operación de convolución sobre los valores u observaciones de entrada antes de pasarlos a una RNAP [62] [63] que haga

una predicción. La convolución se expresa así:

$$Conv(t) = (I * K)(m, n) = \sum_a \sum_b I(a, b)K(m - a, n - b)$$

Donde m, n son las dimensiones de una cuadrícula o tabla de valores y a, b son las dimensiones del *kernel* o *filtro*, que es el área, ventana o subcuadrícula en donde se aplicará una convolución. La convolución toma subcuadrículas de la cuadrícula original y a cada una le aplica un producto punto con el kernel, sumando los resultados de esos productos para reducir sus valores a un único valor, que representa la similitud entre el kernel y la subcuadrícula operada.

Por ejemplo, si el kernel tiene una una dimensión 2×2 con cuatro pesos, hará la convolución sobre una subcuadrícula de 2×2 con cuatro valores. Cada peso multiplica al valor de su misma posición, generando cuatro valores que se terminan sumando para producir un único valor que es el resultado de la convolución.

El kernel inicia aplicando la convolución en la esquina de una cuadrícula (la superior derecha por ejemplo), y después se recorre horizontalmente una unidad para hacer otra convolución. Cuando el kernel llega a la otra esquina (superior izquierda), vuelve a la esquina inicial y se desplaza una unidad verticalmente (hacia abajo), para repetir el proceso hasta llegar la última esquina (inferior izquierda). El número de casillas que se desplaza el kernel horizontal o verticalmente puede ser cambiado con el propósito de obtener mejores resultados.

La operación de convolución es equiparable a una RNAP, así que los valores del kernel se ajustan en cada época o iteración de una RNC con el Algoritmo de Propagación Hacia Atrás para lograr una mayor capacidad de predicción. En el caso de cuadrículas donde en cada casilla hay valores con más de una dimensión, como los valores RGB de los pixeles en una imagen, la convolución devolverá un conjunto de valores con una dimensión mayor a la cuadrícula original.

3.3. Extracción de características

La extracción de características consiste en aplicar algún algoritmo sobre los datos que se procesarán con el fin de encontrar información más precisa de ellos, lo que puede mejorar los resultados de los modelos de Aprendizaje Automatizado. Además ofrece otras ventajas como agilizar el proceso de cómputo, filtrar información irrelevante o redundante, ahorro de memoria de almacenamiento y extracción del ruido en señales [38].

Para este trabajo se decidió aplicar una extracción de características enfocada a obtener nuevas variables respecto de las originales, con una reducción de dimensionalidad que mantenga la relevancia de la información y agilice la ejecución de los algoritmos. Se usaron los algoritmos de Análisis de Componentes Principales y Análisis Discriminante Lineal, que son algunos de los preferidos al extraer características [64] [65].

3.3.1. Análisis de Componentes Principales (ACP)

ACP se define como un método estadístico multivariado no supervisado, el cual tiene buen desempeño y es de los más utilizados [66]. Proyecta la información de determinadas variables a una cantidad reducida de variables, conocidas como *componentes principales* (CP), lo que permite disminuir el espacio de memoria utilizado al convertir la información en un espacio de menor número de dimensiones. Los CP permiten representar datos de una forma compacta y aproximada [67].

Para generar los CP se busca una combinación lineal de las variables que alcance la mayor varianza posible [68]. Los CP no deben tener correlación con los demás. Por ejemplo, si se tienen las variables x_1, x_2, x_3 y x_4, un componente principal podría ser:

$$CP_1 = 0.45x_1 + 0.783x_2 - 0.356x_3 + 0.5x_4$$

El coeficiente de cada variable en la combinación lineal indica qué tanto peso o relevancia tiene para las observaciones; Un coeficiente más alto significa que una variable tiene más varianza, por lo que es más adecuada para representar los datos por sí misma a comparación de otras variables. Para que los resultados de ACP sean correctos se recomienda estandarizar las variables de manera que cada una tenga una media de 0 y una varianza de 1.

Una manera de producir los CP es formando una matriz de covarianza a partir de la matriz de observaciones originales y a esta aplicarle una *Descomposición de Eigenvalores*. Los Eigenvalores se ordenan desde el mayor al menor, representando el primero el componente principal con mayor varianza (CP_1) y el último el que tiene menos (CP_n). Cada CP puede expresarse o graficarse como un eje, que es ortogonal a los demás CPs o ejes, representando un espacio donde la varianza en los datos es máxima.

El número de CPs que se utilizan se define según el porcentaje de varianza que se desea conservar respecto a los datos originales. La representación gráfica de los CP hace posible observar con mayor facilidad comportamientos en las variables que se estén estudiando [69].

Se aplicó ACP para diagnosticar la retinopatía diabética utilizando 19 variables predictoras formadas por medidas de la retina, detección de daños visibles en la retina y un valor binario perteneciente a una clasificación de amplitud y frecuencia modulada. Se obtuvo una exactitud de $96\,\%$ para los datos sin PCA y $90.07\,\%$ con PCA utilizando Redes Neuronales Profundas, $57\,\%$ y $63.5\,\%$ para Bayes Ingenuo, $73.8\,\%$ y $65.3\,\%$ con Máquinas de Soporte Vectorial y, $83.3\,\%$ y $80.0\,\%$ usando XGBoost [70]. Como se observa, sólo mejoran los resultados para Bayes Ingenuo.

3.3.2. Análisis Discriminante Lineal (ADL)

ADL surge a partir de la teoría probabilística aplicada a las variables predictoras de las observaciones de entrenamiento, con la cual se obtiene una función lineal que separa los datos en clases o variables objetivo. Para producir esa función, el algoritmo necesita conocer de antemano

los valores de las variables objetivo de las observaciones de entrenamiento, por lo que se dice que es un método de aprendizaje automatizado supervisado.

ADL es un método de análisis estadístico multivariado que hace una transformación lineal de las variables predictoras de forma que las observaciones que tengan los mismos valores de variables objetivo sean proyectadas en un espacio donde queden cercanas unas de otras, lo que agrupa o separa las observaciones según los valores de variables objetivo que existan [66] [71]. El algoritmo supone que los datos que procesa tienen una distribución normal, donde cada observación tiene una matriz de covarianza igual a las demás y todas son estadísticamente independientes entre sí. Este método también permite disminuir el sobreajuste provocado por un exceso de dimensiones en los datos. Para poder aplicar el algoritmo, las variables predictoras necesitan estar previamente estandarizadas.

ADL se define matemáticamente como lo siguiente [72][73]:

Se obtiene un vector para cada valor posible de clase o variable objetivo. En cada vector están las medias de cada variable predictora x_n pertenecientes a las observaciones que tengan el valor de variable objetivo del vector al que corresponden.

$$m_i = \frac{1}{n_i}\sum_{n=1}^{n_i} x_n = \begin{bmatrix} \mu_i 1 \\ \vdots \\ \mu_i n \end{bmatrix}^T$$

Donde i es un número que identifica a un valor posible de variable objetivo, n es un valor que identifica a una variable predictora y μ es el promedio de los valores de una variable predictora. A partir de estos vectores se calcula la matriz de dispersión dentro de las variables objetivo S_W.

$$S_W = \sum_{i=1}^{v_o} S_i$$

v_o es el número total de valores posible de variables objetivo. S_i es la *matriz de dispersión de cada valor posible de variable objetivo*.

$$S_i = \sum_{n=1}^{n_i} (x_n - m_i)(x_n - m_i)^T$$

Después se normaliza S_i, lo que la hace equivalente a la matriz de covarianza Σ_i.

$$\Sigma_i = \frac{1}{n_i} S_i = \frac{1}{n_i}\sum_{n=1}^{n_i} (x_n - m_i)(x_n - m_i)^T$$

Posteriormente se hace la *matriz de dispersión entre los posibles valores de variables objetivo* S_B.

$$\Sigma_B = \sum_{i=1}^{v_o} n_i (m_i - m)(m_i - m)^T$$

En la cual m es el vector de medias de cada variable predictora de todas las observaciones, sin hacer distinción según los valores de su variable objetivo. Con las matrices anteriores se generan

eigenvalores y eigenvectores a partir de $S_W^{-1}S_B$. Los eigenvalores se ordenan de mayor a menor para conocer la relevancia de sus eigenvectores. Cada eigenvector es un discriminante lineal con mayor o menor relevancia según la posición en orden descendente de sus eigenvalor. Por la naturaleza de las operaciones matriciales utilizadas, del total de eigenvalores sólo $v_o - 1$ deben tener un valor que no sea 0, o no muy cercano a 0. Finalmente, para proyectar las observaciones a un nuevo espacio donde los datos estén ubicados en sectores diferentes según el valor de variable objetivo que tienen, se forma una matriz W con los $v_o - 1$ eigenvectores que no tienen un valor de 0 en sus eigenvalores, la cual multiplica a la matriz de observaciones X.

$$X' = XW$$

3.4. Métricas de Evaluación

En el caso de los algoritmos que procesan observaciones con sólo una variable objetivo de valor booleano o verdadero y falso, las métricas que describen su desempeño se obtienen a partir de los verdaderos positivos (VP), falsos positivos (FP), verdaderos negativos (VN) y falsos negativos (FN). Las métricas que se describen a continuación son de las más utilizadas al evaluar algoritmos de Aprendizaje Automatizado con las características mencionadas [70].

Exactitud

Indica el porcentaje de observaciones que se clasificaron correctamente. Se expresa como:

$$Exactitud = \frac{VP + VN}{VP + VN + FP + FN}$$

Precisión

Determina el porcentaje de observaciones que se clasificaron como VP del total de positivos detectados. Con ello se conoce qué también se predicen las observaciones que son positivas según todas las observaciones positivas detectadas correcta e incorrectamente. Se obtiene con:

$$Precisión = \frac{VP}{VP + FP}$$

Sensibilidad

Es el porcentaje de VP que se clasificaron correctamente, indicando la capacidad para clasificar observaciones positivas respecto de todas las observaciones que verdaderamente eran positivas. Es calculada mediante la siguiente fórmula:

$$Sensibilidad = \frac{VP}{VP + FN}$$

Especificidad

Porcentaje de VN que se clasificaron correctamente del total de observaciones negativas existentes. Representa el desempeño para detectar observaciones negativas:

$$Especificidad = \frac{VN}{VN + FP}$$

F1-Score

Es una medida de la capacidad de un modelo para clasificar los VP según un promedio harmónico calculado a partir del porcentaje de VP obtenidos respecto a todos los positivos detectados y del porcentaje de VP clasificados del total de positivos que en realidad existían en las observaciones clasificadas. Se puede decir que es una forma más general de observar la capacidad de detectar VP utilizando la precisión y sensibilidad al mismo tiempo:

$$F1Score = \frac{2 * (\frac{VP}{VP+FP}) * (\frac{VP}{VP+FN})}{(\frac{VP}{VP+FP}) + (\frac{VP}{VP+FN})}$$

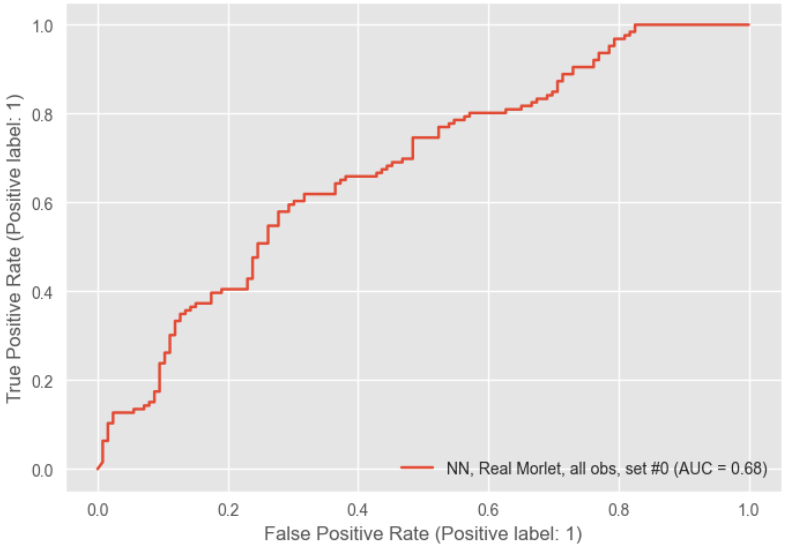


Figura 3.4: Ejemplo de ABC COR

ABC COR (Área Bajo la Curva de la Característica Operativa del Receptor)

Esta métrica permite analizar el desempeño de un modelo al comparar la sensibilidad y la especificidad en una gráfica de dos dimensiones. En el eje vertical se tiene el valor de la sensibilidad y en el eje horizontal el inverso de la especificidad ($1-$ especificidad). Al graficar esta relación se traza una curva de la que se calcula el área debajo de ella, ya que este valor representa la capacidad de clasificar VP y VN respecto a todas las observaciones.

Cuando se obtiene un área de 0.5 formada por una línea diagonal, significa que un modelo no puede distinguir entre positivos y negativos. Si el valor es 0 el modelo está clasificando invertidamente los positivos y negativos. Mientras el valor se acerque más a 1, un modelo será mejor para diferenciar entre positivos y negativos, convirtiéndose en un buen clasificador. El ABC COR (Figura 3.4) tiene la cualidad de mantener valores similares a la exactitud, sirviendo como una medida alterna o de comparación para esta métrica [74].

3.5. Optimización de Hiperparámetros

Un hiperparámetro es una variable utilizada por un algoritmo de Aprendizaje Automatizado, la cual debe tener un valor inicial para la ejecución del algoritmo. El valor de un hiperparámetro puede influir en gran manera en el desempeño, resultados y tiempo de procesamiento del algoritmo, por lo que debe haber una metodología para definir el valor más adecuado de cada hiperparámetro. La optimización de hiperparámetros se define como la búsqueda de los hiperparámetros para determinado modelo que produzcan el mejor desempeño o el mejor valor posible de alguna métrica, como exactitud, precisión, etcétera. El que una persona pruebe manualmente distintas combinaciones de hiperparámetros no es óptimo y puede provocar sesgo, por lo que se recomienda utilizar algoritmos automáticos que prueben los hiperparámetros [45] [75].

Para la optimización de hiperparámetros de los modelos utilizados en este estudio, se ocupó la plataforma de uso libre Optuna [76]. Una de las ventajas de esta plataforma a comparación de otras es que se puede importar como una biblioteca más de Python, siguiendo la sintaxis y estilo del lenguaje, lo que la hace más fácil e intuitiva de utilizar. Además tiene una sintaxis que permite declarar en una función los hiperparámetros que se van a optimizar y el intervalo y tipo de valores que se aplicarán [77].

Optuna funciona por medio de una etapa de *muestreo* y una etapa de *poda*. En el muestreo se combina el algoritmo de *Optimización Bayesiana TPE* con el *CMA-es*. La Optimización Bayesiana se define como un algoritmo iterativo con dos fases por iteración, un *modelo de sustitución probabilística* y una *función de adquisión*.

La primera fase usa un proceso gaussiano para formar una distribución de probabilidad de los valores de una *función objetivo* (una función o modelo de Aprendizaje Supervisado que produce la métrica que se quiere mejorar). Esta distribución se construye al tomar varias muestras de la métrica producida probando hiperparámetros iniciales aleatorios. La segunda fase usa esa distribución para decidir qué valores de hiperparámetros aplicar de manera que se produzca el mejor valor de métrica. A partir de la segunda iteración se actualiza la distribución de probabilidad con el valor de métrica resultante de probar los hiperparámetros decididos por la función de adquisición anterior, y con ella se calcula otra función de adquisición [78].

CMA-es (Estrategia de evolución para la Adaptación de la Matriz de Covarianza) también es un algoritmo iterativo, pero que actualiza los hiperparámetros según una MGD (Distribución Gaussiana Multivariada) [79]. El usuario puede configurar sus propios algoritmos de *Automatización de Búsqueda de Hiperparámetros* si no desea los anteriores, que son los que están establecidos por defecto.

La poda es la detención de una iteración de la optimización de hiperparámetros antes de que termine si se detecta que los hiperparámetros que se están probando producen alguna métrica con poco desempeño. Esta etapa es aplicable si el modelo de Aprendizaje Automatizado cuenta con iteraciones o *épocas* internas que devuelvan un valor de los que se pueda obtener una *curva de aprendizaje* para evaluar.

3.6. Transformada Continua de Wavelet (TCW)

Al aplicar un procesamiento o transformación matemática a un ERG se pueden revelar características de la señal que facilitan y mejoran la identificación de patologías oculares. En las frecuencias bajas del rango $15 - 40[Hz]$ se encuentran las ondas a (positivas) y b (negativas) fotópicas, que son de las más análizadas en estudios de ERG. Este tipo de señales puede ser utilizada en los dominios del tiempo, frecuencia, tiempo-frecuencia y con métodos no lineales; Al usar un dominio de tiempo-frecuencia se obtiene una interpretación y entendimiento más amplios. El análisis con wavelets es uno de los métodos tiempo-frecuencia más utilizados, ya que es adecuado para señales no estacionarias como los ERG [80].

Este tipo de análisis se puede aplicar con la Transformada Continua de Wavelet (TCW) o con la Transformada Discreta de Wavelet (TDW). Para esta investigación se usó la TCW, ya que esta operación matemática tiene la propiedad de filtrar frecuencias de interés en una señal. Además tiene alta resolucíon de frecuencias para frecuencias bajas, como las de este estudio. Su desventaja es que debe encontrarse la wavelet madre más adecuada para una buena extracción de características. Se expresa como lo siguiente:

$$C(\alpha, \tau) = \int \frac{1}{\sqrt{a}} \Psi(\frac{t-\tau}{a}) x(t) dt$$

Donde a es un factor de escala, τ es el tiempo de traslación, Ψ es la wavelet madre y $x(t)$ es la señal de entrada en función del tiempo. Cuando $a > 1$ la transformada ofrece una mayor resolución sobre el tiempo, mostrando con mayor detalle fenómenos transitorios. Si $a < 1$, hay una mayor resolución sobre el espectro de potencia, siendo mejor para frecuencias de fenómenos de estado estacionario [81].

Se dice que no hay una manera formal o precisa de elegir una wavelet madre para procesar un ERG, pero hay estudios que demuestran que la familia *Morlet* es la más adecuada debido a que su forma es parecida a la de un ERG [82]. La mayoría de los métodos de selección de wavelet madre parten de elegir la que se parezca más a la señal que se va a transformar, pero se recomienda elegir la que dé los mejores resultados sin importar las semejanzas con la señal original [83], lo que implica hacer una experimentación con varias wavelets para determinar cuál es más beneficiosa en determinado estudio.

Los datos preprocesados disponibles para este trabajo tienen aplicada la TCW con la wavelet *Morlet* en su versión real y en su versión compleja. Además se tienen los datos preprocesados con TCW aplicando las wavelet madre *Gaussiana Real*, *Gaussiana Compleja*, *Shannon* y *Sombrero Mexicano*. Los algoritmos de Aprendizaje Automatizado y extracción de características se ejecutarán con las 6 versiones para comparar su desempeño y resultados.

3.6.1. Promedio de Potencia en Determinadas Frecuencias

La Transformada de Wavelet permite formar un *escalograma*, que es una representación de la potencia en determinada frecuencia a lo largo del tiempo. A partir de ese escalograma se obtuvo el promedio de la potencia en el minuto de duración de la señal ERG para cada una de 71 frecuencias en el intervalo de interés mencionado anteriormente, seleccionadas de forma logarítmica. Cada uno de esos promedios de frecuencias es una variable predictora en los modelos de Aprendizaje Automatizado implementados.

Capítulo 4

Descripción, análisis y preparación de los datos

4.1. Características de los datos

Se tuvo acceso al conjunto de información utilizado gracias a la participación en un proyecto conformado por integrantes de la Facultad de Ingeniería de la UNAM (FI-UNAM) y el Instituto de Neurobiología de la UNAM (INB-UNAM). Los datos consisten en electrorretinogramas (ERG) no fotópicos obtenidos de pacientes sanos y pacientes diagnosticados con retinopatia diabética, los cuales fueron proveídos por las siguientes instituciones: Asociación Para Evitar la Ceguera (APEC), Instituto de Oftalmología, I.A.P. (IMO), Clínica de Salud Visual de ENES-UNAM Unidad León y el Instituto de la Retina del Bajío (INDEREB). La asignación del estado sano o enfermo para cada uno de los pacientes fue realizada por personal de salud experto en el área.

4.2. Preprocesamiento

Los datos usados en este trabajo ya tenían aplicado un preprocesamiento previo. La aportación de este estudio al proyecto ya mencionado fue la implementación de los algoritmos de clasificación presentados en el marco teórico, además de la extracción de características con ACP, ADL y la optimización de hiperparámetros. A continuación se describen las características de este preprocesamiento previo y su justificación.

Las medidas de todas las clínicas tuvieron una frecuencia de muestreo de $1000[Hz]$, expresándose en $[mV]$. Los datos fueron recopilados en archivos *.csv*. De esos archivos se extrajeron las columnas que tenían los voltajes de cada ERG, el número de paciente al que pertenecían y si es un enfermo o sano. Algunos segundos iniciales y finales de cada ERG fueron eliminados, al ser ruido provocado por la colocación y retiro de los electrodos que se usan para medir un ERG. A los datos restantes se les aplicó una normalización *minmax* con un intervalo de -1000 a 1000.

Cada clínica o institución utilizó diferentes equipos para recopilar los ERG, con un periodo de tiempo diferente, por lo que el ERG de cada paciente se dividió en fragmentos de 60 segundos, generando varias observaciones por cada paciente.

Posteriormente, a cada observación se le aplicó una *Transformada Continua de Wavelet*, con el fin de obtener el espectro de potencia promedio en el intervalo $0.4[Hz]$ a $40[Hz]$, considerado el más adecuado para revelar enfermedades en la retina. Para probar el efecto de esta operación con más detalle, se usaron seis variantes, cada una con una *wavelet madre* diferente (Morlet Compleja, Morlet Real, Gaussiana Compleja, Gaussiana Real, Shannon y Sombrero mexicano).

Como resultado del preprocesamiento, cada observación tiene como variables el número de paciente (identificador), estado de salud (variable objetivo) y los 71 valores promedio de potencia generados.

Capítulo 5

Implementación de algoritmos de clasificación

5.1. Consideraciones generales

Después de aplicar el preprocesamiento en total se formaron 632 observaciones de 98 pacientes sanos y 2250 observaciones de 357 pacientes enfermos para cada wavelet. Como se puede apreciar, el número de pacientes sanos es menor, por lo que los conjuntos de entrenamiento y prueba se crearon de manera que se aprovechen todos los pacientes sanos.

Al aplicar un balanceo de datos (mitad enfermos y mitad sanos) además de una distribución $80/20$ ($80\,\%$ datos para entrenamiento y $20\,\%$ datos para prueba), el conjunto de entrenamiento se conformó con 506 observaciones de un número aleatorio de pacientes sanos y 506 observaciones de un número aleatorio de pacientes enfermos. A su vez, el conjunto de prueba contó con 126 observaciones de un número aleatorio de pacientes sanos y 126 observaciones de un número aleatorio de pacientes enfermos. Cabe recordar que cada paciente tiene una cantidad distinta de observaciones, lo que provoca que para alcanzar el total de 506 o 126 observaciones, el total de pacientes utilizados sea aleatorio en cada conjunto.

Además, se condicionó colocar todas las observaciones de un paciente sólo en el conjunto de entrenamiento o sólo en el de prueba, no en ambos, para asegurar que no exista un sesgo por colocar datos de un paciente tanto en el entrenamiento como en la prueba.

Para comprobar con mayor rigor estadístico el desempeño de los modelos utilizados con los datos originales y con los procesados a través de ACP y ADL, se formaron aleatoriamente 10 conjuntos de observaciones para cada wavelet (un conjunto tiene los subconjuntos de entrenamiento y prueba). A cada uno se le aplicó ACP y ADL, con lo que se acumulan 30 conjuntos por wavelet (10 originales, 10 ACP y 10 ADL). Algunos modelos no fueron aptos para recibir los datos con ACP y ADL por razones que serán explicadas más adelante, así que en el caso de los que sí lo fueron, se ejecutaron 30 veces por wavelet.

Por tener 6 tipos de wavelet, los conjuntos totales son 180. El total de experimentos o

ejecuciones de modelos para este trabajo considerando que algunos modelos no se implementaron con ACP y ADL, fue de 360 con los datos originales, 300 con ACP y 240 para ADL.

La ejecución de los modelos, preprocesamiento, optimización de hiperparámetros y la extracción de características se codificaron en el lenguaje de programación Python, al ser uno de los que disponen más bibliotecas y documentación enfocadas al Aprendizaje Automatizado. La versión utilizada en todo el código realizado fue la 3.9.

Los algoritmos de Redes Neuronales Convolucionales y Redes Neuronales Recurrentes se implementaron con la biblioteca Keras y el resto de los algoritmos con la biblioteca Scikit-learn. La ventaja de Keras es que se pueden diseñar modelos de Redes Neuronales y Aprendizaje Profundo con un sistema de capas que permite personalizar ágilmente la estructura, parámetros y tipo de las capas de una red. Por otro lado, Scikit-learn hace posible implementar los modelos de Aprendizaje Automatizado más conocidos con funciones similares, lo que ayuda a la reutilización de código. También integra métodos de evaluación de modelos y otras herramientas útiles para el Aprendizaje Automatizado; La evaluación de métricas y la obtención de gráficas se realizó con bibliotecas de Scikit-learn.

Almacenamiento de Resultados

Los resultados fueron ordenados y recopilados a través de la plataforma *Neptune*, que facilita la visualización y control de las métricas u otra información relevante de los modelos de Aprendizaje Automatizado que se estén estudiando, guardándola en sus servidores. Neptune es gratis en su versión simple y se puede acceder a él desde un navegador de internet. Además Neptune es implementado con lenguaje Python, lo que agiliza su integración al código utilizado en este trabajo.

Optimización de Hiperparámetros

La optimización se enfocó en mejorar la métrica de exactitud, siendo la que representa las observaciones que se clasificaron correctamente. Se utilizaron 100 iteraciones para las Redes Neuronales, 50 para las Redes Neuronales Convolucionales y Recurrentes, y 2000 para el resto de algoritmos. El número de iteraciones del algoritmo de optimización de hiperparámetros fue establecido según una gráfica de rendimiento (Figura 5.1), que muestra el valor de exactitud obtenido en cada iteración. La curva formadada en la gráfica permite observar si las iteraciones han llegado a un límite de optimización o si hay posibilidad de obtener mejor exactitud al incrementar las iteraciones. Al revisar las gráficas se constató que el número de iteraciones era suficiente para alcanzar el límite de optimización en la mayoría de los experimentos. Se decidió mantener la misma cantidad de iteraciones en determinado modelo con el objetivo de comparar su desempeño para cada wavelet y con cada forma de extracción de características en igualdad de condiciones.

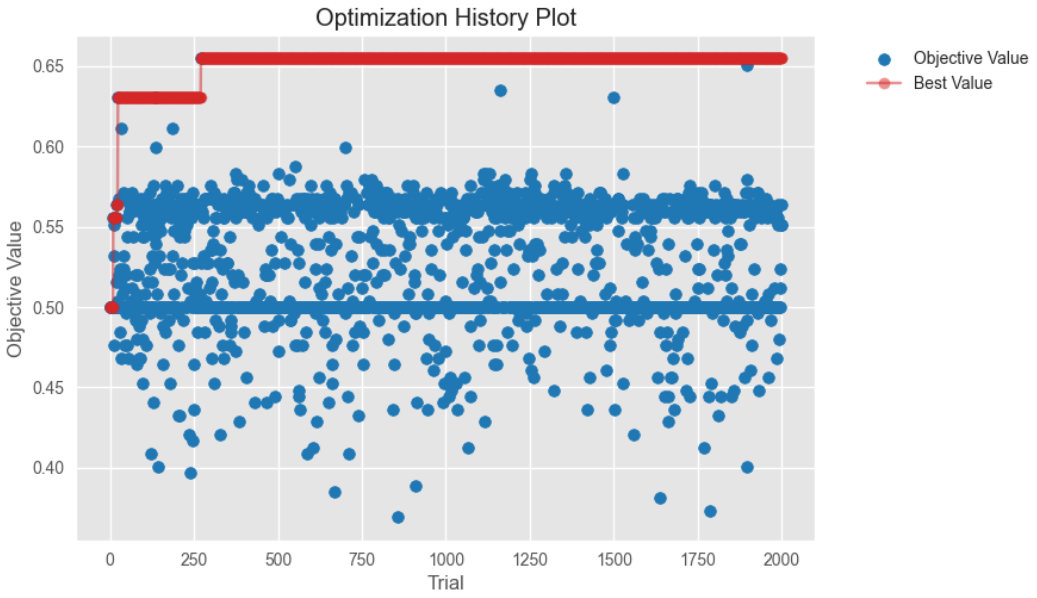


Figura 5.1: Gráfica de la exactitud obtenida en las iteraciones de la optimización de hiperparámetros para Bosques Aleatorios con las variables predictoras originales producidas con la wavelet Gaussiana Compleja. El eje vertical marca los valores de exactitud y el eje horizontal la iteración. La línea roja marca el valor más alto de exactitud.

Para conocer si los hiperparámetros originados tienen el mismo rendimiento con cualquier conjunto aleatorio de datos, se optimizaron los 6 modelos de Aprendizaje Automatizado usando sólo un conjunto de los 10 generados para cada wavelet y cada tipo de extracción de características (datos originales, ACP y ADL). Los hiperparámetros resultantes de optimizar determinado conjunto y modelo se utilizaron para ejecutar ese modelo con los 10 conjuntos.

Los rangos de valores probados en los hiperparámetros de arquitectura o forma de todos los tipos de redes neuronales se mantuvieron con valores bajos, ya que no hay una cantidad de datos amplia para aplicar un Aprendizaje Profundo con muchas capas. La ventana de convolución también se mantuvo pequeña al tener observaciones de $1X71$ variables predictoras.

El resto de hiperparámetros en todos los modelos se colocaron con los máximos y mínimos valores posibles, con todos los valores existentes (en el caso de valores discretos) o con un rango de valores elegido arbitrariamente, ya que no en todos los hiperparámetros hay una regla o manera precisa de elegir intervalos de valores. Los hiperparámetros obtenidos en la optimización se muestran en el *Apéndice B* y las gráficas de cada optimización se encuentran en el *Apéndice C*.

5.2. Consideraciones en cada algoritmo

Bosques Aleatorios

Creado a partir de la clase *sklearn.ensemble.RandomForestClassifier*. Los hiperparámetros optimizados fueron *random_state*, un número entero que controla la aleatoriedad al dividir las observaciones para formar árboles y las observaciones para probar los nodos de estos,

n_estimators para indicar la cantidad de árboles utilizados en el algoritmo, *max_samples*, el cual es el número de observaciones con las que se entrena cada árbol, *min_samples_leaf*, que define el total de observaciones mínimas para formar un nodo hoja y *min_samples_split* con el propósito de establecer el número de observaciones mínimas para dividir un nodo y formar otros.

Bayes Ingenuo Multinomial

Se formó con la clase *sklearn.naive_bayes.MultinomialNB*. En este caso la optimización se enfocó en los parámetros *fit_prior*, que es una variable booleana para decidir si se calculan las probabilidades *a priori* o si no se calculan y se toma en cuenta una distribución de probabilidad uniforme, y *alpha*, que es un valor flotante del *Suavizado de Laplace*, una técnica para ajustar una probabilidad en caso de que sea igual a 0, lo que evita errores en el algoritmo.

Este algoritmo no es compatible con valores negativos al considerar una distribución multinomial que es positiva, por lo que ACP y ADL no fueron probados en él.

Bayes Ingenuo Gaussiano

Implementado con la clase *sklearn.naive_bayes.GaussianNB*. El único hiperparámetro optimizado es *var_smoothing*, un valor flotante que se agrega a las varianzas para mantener la estabilidad de los cálculos del algoritmo. Este número es una porción de la varianza más alta.

Redes Neuronales

Las capas de esta red neuronal simple fueron establecidas con la clase *sklearn.neural_network .MLPClassifier*. Existe una gran variedad de arquitecturas y condiciones para armar una red neuronal de cualquier tipo. Para esta investigacion se probaron redes neuronales con una configuración simple, con el fin de limitar el objetivo de estudio.

Se optimizó el hiperparámetro *n_layers*, que representa el número de capas de la red neuronal. Para cada una de las capas que se hayan establecido con *n_layers*, se optimizó *n_units*, que es la cantidad de perceptrones en la capa. También se optimizó la función de activación de los perceptrones, probando la *identidad*, *tangente hiperbólica*, *logística* y *relu*. A su vez el *optimizador* de la función que calcula el error en la red neuronal fue probado con las funciones *L-BFGS*, *SGD* (Gradiente Descendente Estocástico) y *Adam*. Además se optimizó el hiperparámetro *alpha* utilizado en la *regularización L2*, que es un valor que se resta a los pesos de la red neuronal para mantener la estabilidad al evitar el sobreajuste. Por último se optimizó *random_state*, un entero que permite colocar valores aleatorios en la *unidad de sesgo* y en los pesos iniciales de la red neuronal. El máximo de *épocas* se mantuvo con el valor constante de 1000 (*max_iter*) y la tolerancia de error o límite de convergencia (*tol*) se fijó en 0.000000001, esto con el objetivo de mantener una igualdad de condiciones en las épocas y convergencia de las redes neuronales examinadas en cada iteración.

Redes Neuronales Convolucionales

Este tipo de red neuronal se construyó a partir de la clase *keras.models.Sequential*, que permite formar una red neuronal agregando capas del tipo que se necesiten, asegurando que las entradas y salidas de las capas sean compatibles entre ellas. Las capas utilizadas en este algoritmo en orden desde la entrada de los datos hasta la salida de la clasificación fueron *keras.layers.Conv1D*, *keras.layers.Flatten* y *keras.layers.Dense*. *Conv1D* aplica el algoritmo de convolución sobre los datos de entrada y el resultado se pasa a una capa *Flatten* (aplanado) que ordena los valores que recibe en un vector de $1Xn$ dimensiones, lo que hace posible introducirlo a una capa *Dense* (Red Neuronal Simple), que devuelve dos valores que representan los estados sano y enfermo. Se usó una capa Conv1D que permite hacer la convolución sobre vectores de $1Xn$ dimensiones, ya que las dimensiones de las variables predictoras de cada observación son $1X71$.

La convolución necesita mínimo 2 valores para aplicarse, por lo que ADL no fue probado con esta red, al tener un valor por observación. Los hiperparámetros optimizados fueron el número de filtros en la convolución *filters*, el tamaño de ventana de convolución *kernel_size*, la función de activación *activation*, la cantidad de casillas que se desplazará la ventana de convolución *strides* y la tasa de aprendizaje del optimizador Adam, *learning_rate*. Se fijaron 100 épocas en la red y una activación *softmax* para la capa de salida *Dense*.

Redes Neuronales Recurrentes

Al igual que la Red Neuronal Convolucional, esta red se configuró con la clase *keras.models.Sequential*, teniendo la misma estructura de capas excepto por la primera, quedando de la siguiente manera: *keras.layers.SimpleRNN*, *keras.layers.Flatten* y *keras.layers.Dense*. *SimpleRNN* aplica el algoritmo de Redes Neuronales Recurrentes. Los hiperparámetros optimizados son la dimension del espacio de salida *units*, la función de activación *activation* y la tasa de aprendizaje del optimizador Adam, *learning_rate*. También se fijaron 100 épocas en la red y una activación *softmax* para la capa de salida *Dense*.

Capítulo 6

Análisis de Resultados

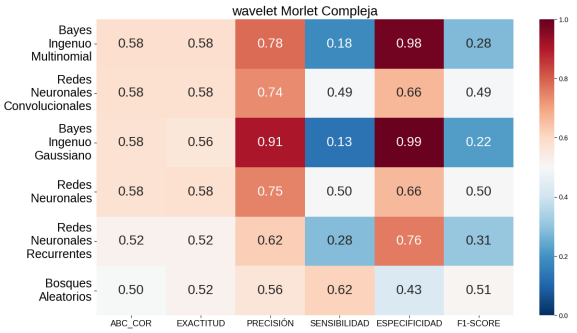
Después de obtener los resultados de los algoritmos de Aprendizaje Automatizado utilizados con una wavelet y un tipo de extracción de características en 10 conjuntos, se promediaron sus métricas y se graficaron en un mapa de calor. Estas gráficas se presentan y analizan a continuación.

Morlet Compleja

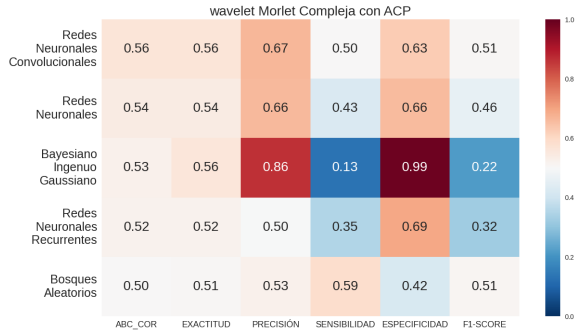
Con los datos originales (Figura 6.1a) los algoritmos bayesianos, la Red Neuronal y la Red Neuronal Convolucional mantienen el mismo valor de ABC_COR (0.58) y valores similares de exactitud $(0.56 - 0.58)$. Los algoritmos bayesianos tienen valores de especificidad altos y de sensibilidad bajos, lo que indica que los pacientes sanos y enfermos son detectados en su mayoría como sanos. La precisión alta de los bayesianos representa que de los pocos pacientes reconocidos como enfermos, la mayoría sí lo están. La f1-score baja en todos los modelos indica una poca capacidad para detectar a los pacientes enfermos.

Todos los tipos de Redes Neuronales siguen el mismo comportamiento de precisión, sensibilidad y especificidad que los algoritmos bayesianos, pero con un poco de mayor capacidad para distinguir entre sanos y enfermos. Los Bosques Aleatorios y las Redes Neuronales tienen una exactitud (0.50) y ABC_COR (0.52) similar a las Redes Neuronales Recurrentes (0.52) , siendo los modelos con peor desempeño. Los Bosques Aleatorios no detectan a los pacientes sanos efectivamente, al tener una especificidad baja.

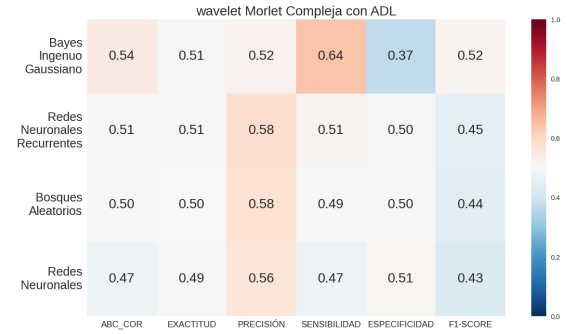
Al utilizar ACP (Figura 6.1b) con los modelos aptos para ello, el comportamiento anterior se mantiene pero disminuyendo su desempeño. Con ADL (Figura 6.1c) se mantiene el mismo comportamiento que con las variables originales, con resultados peores que ACP. Se alcanza un mayor equilibrio entre los valores de las métricas, excepto para Bayes Ingenuo Gaussiano, que comienza a distinguir un poco mejor a los enfermos por subir su sensibilidad y f1-score. Los modelos que suben su f1-score, lo hacen al mismo tiempo que disminuyen su precisión y especificidad.



(a) Variables originales



(b) ACP



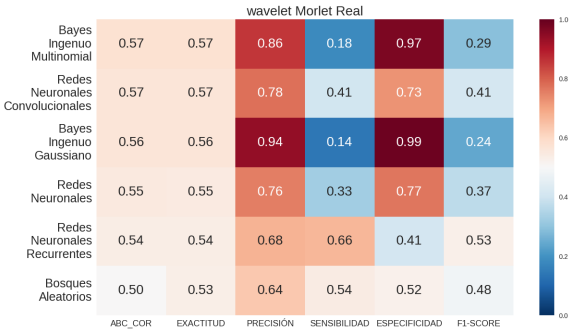
(c) ADL

Figura 6.1: Resultados con la wavelet Morlet Compleja

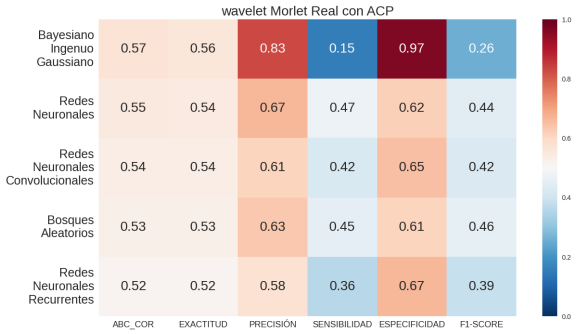
Morlet Real

Para la wavelet Morlet Real con los datos originales (Figura 6.2a), los modelos siguen el mismo comportamiento que la Morlet Compleja y mantienen el mismo orden desde el mejor a el peor ABC_COR, con la diferencia que las Redes Neuronales Recurrentes invierten su comportamiento y clasifican mejor a los enfermos que a los sanos. Bosques Aleatorios detecta mejor a los enfermos pero distingue peor a los sanos. F1-score con valor bajo en todos los modelos.

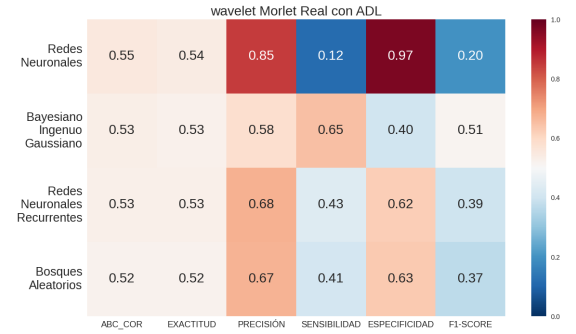
Al aplicar ACP sobre la Morlet Real (Figura 6.2b), Bayes Ingenuo Gaussiano empeora ligeramente sus métricas, excepto por el ABC_COR, la exactitud, sensibilidad y f1-score. Bosques Aleatorios mejora un poco su ABC_COR y mantiene su exactitud. Las Redes Neuronales Recurrentes aumentan su especificidad y disminuye su sensibilidad, lo que lo hace confundir enfermos con sanos. El resto de los comportamientos se mantiene pero con peores valores de métricas. F1 se incrementa en Bayes Ingenuo Gaussiano, Redes Neuronales y las Redes Neuronales Convolucionales. Con ADL (Figura 6.2c) las métricas empeoran y se equilibran, excepto en las Redes Neuronales, donde sus valores de especificidad altos con f1-score baja muestran que se clasifican la mayoría de pacientes como sanos. La F1-score sólo se incrementa en Bayes Ingenuo Gaussiano.



(a) Variables originales



(b) ACP



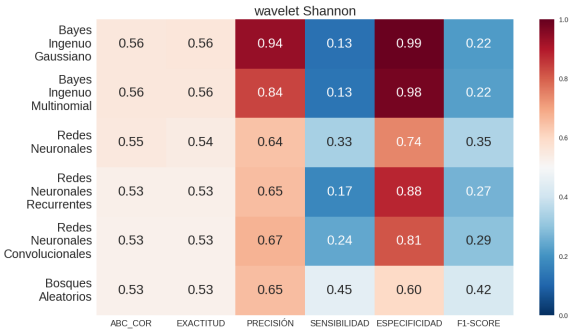
(c) ADL

Figura 6.2: Resultados con la wavelet Morlet Real

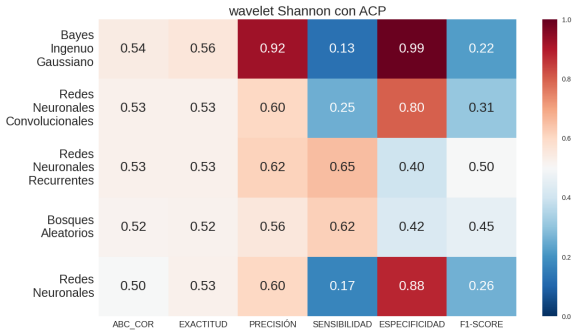
Shannon

Al utilizar todas las variables originales con la wavelet Shannon (Figura 6.3a), todos los modelos presentan una precisión y especificidad altas o mayores en comparación con su sensibilidad, teniendo el mismo comportamiento que la wavelet Morlet Compleja donde la mayoría de las observaciones se marcaban como pacientes sanos y los pocos pacientes detectados como enfermos sí lo estaban. Todos los modelos tienen dificultad para detectar a los pacientes enfermos, como lo indican sus f1-score bajas.

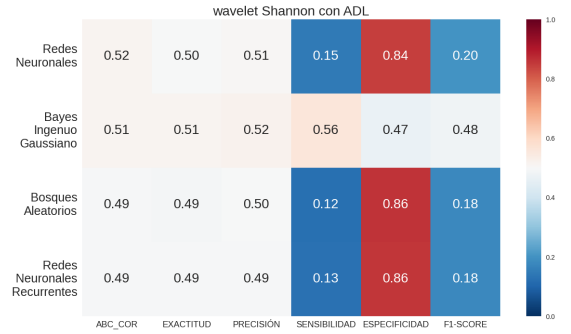
Con ACP (Figura 6.3b) las Redes Neuronales Recurrentes y las Redes Neuronales Convolucionales mantienen sus valores de exactitud y ABC_COR, el resto de los modelos reducen su efectividad en estas métricas. En cuanto a la precisión, sensibilidad y especificidad, Bosques Aleatorios y Redes Neuronales Recurrentes invierten su comportamiento marcando a la mayoría de pacientes como enfermos. F1-score sube para Redes Neuronales Recurrentes, Redes Neuronales Convolucionales y Bosques Aleatorios, baja para Redes Neuronales, y se mantiene en Bayes Ingenuo Gaussiano.



(a) Variables originales



(b) ACP

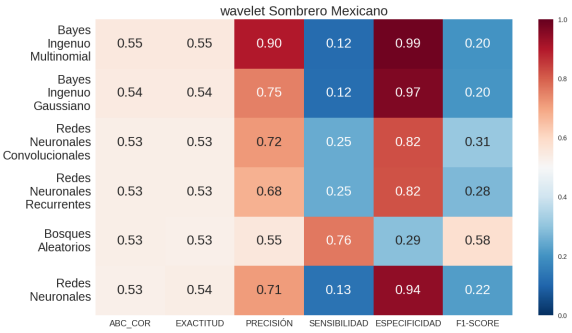


(c) ADL

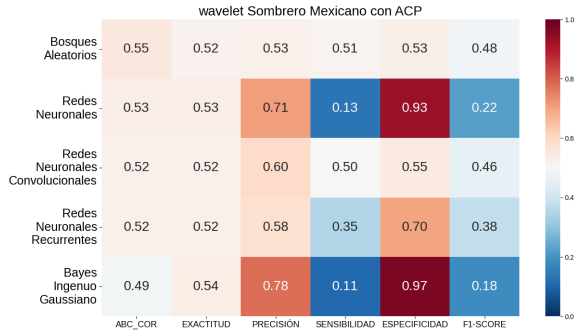
Figura 6.3: Resultados con la wavelet Shannon

En ADL (Figura 6.3c) todas las métricas disminuyen a comparación de hacer uso de las variables originales.

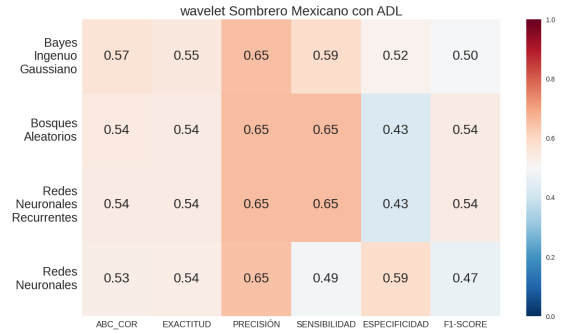
La poca sensibilidad y alta especificidad se mantienen, pero la precisión disminuye. Lo primero muestra que la mayoría de los pacientes se clasifican como sanos, y lo segundo que tienden a clasificarse enfermos como sanos en mayor medida a comparación de las variables originales. Para Bayes Ingenuo Gaussiano se equilibra la sensibilidad y especificidad, además que aumenta su f1-score. Para el resto de modelos empeora su f1-score.



(a) Variables originales



(b) ACP



(c) ADL

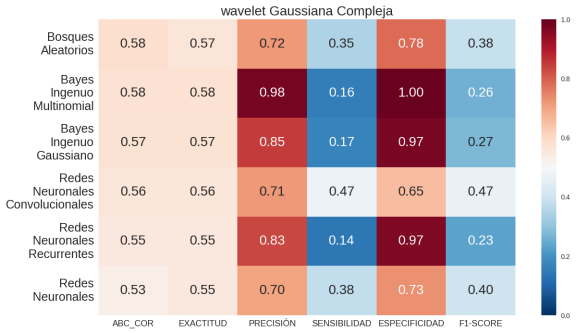
Figura 6.4: Resultados con la wavelet Sombrero Mexicano

Sombrero Mexicano

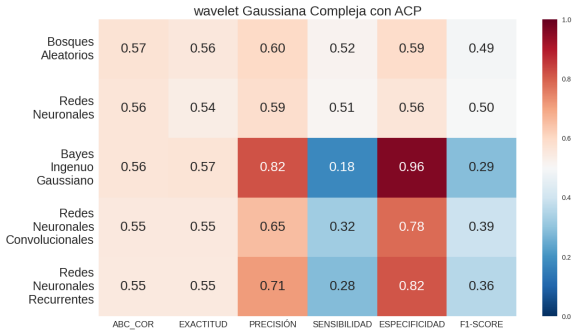
Haciendo uso de los datos originales (Figura 6.4a), esta wavelet también tiene el mismo comportamiento de clasificar a la mayoría de los pacientes como sanos y que los pocos pacientes que detecta como enfermos sí son enfermos en su mayoría. Bosques Aleatorios sigue un patrón contrario y clasifica a la mayoría de los pacientes como enfermos, además que tiene un f1-score mayor en comparación a los demás.

En el caso de ACP (Figura 6.4b) Bosques Aleatorios mejora un poco su capacidad para diferenciar sanos y enfermos, equilibrando sus métricas. Las Redes Neuronales mantienen los mismos valores de métricas, excepto por la especificidad y exactitud que se reducen un poco. Bayes Ingenuo Gaussiano mantiene su comportamiento con una reducción de sus métricas. Las Redes Neuronales Recurrentes y las Redes Neuronales Convolucionales reducen ligeramente su exactitud y ABC_COR a cambio de ganar un mayor equilibrio en las otras métricas, lo que implica una menor confusión de enfermos con sanos y sanos con enfermos. La f1-score aumenta en los modelos que equilibran sus métricas.

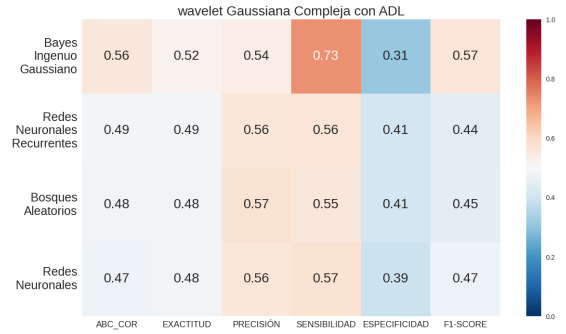
Las Redes Neuronales mantienen su exactitud y ABC_COR con la aplicación de ADL (Figura 6.4a), y obtienen mayor equilibrio en el resto de sus métricas. Los otros modelos mejoran todas sus métricas y adquieren más equilibrio. A comparación de las otras wavelet, ACP y ADL son más efectivos con Sombrero Mexicano, mejorando ligeramente o manteniendo los valores respecto de las variables originales.



(a) Variables originales



(b) ACP



(c) ADL

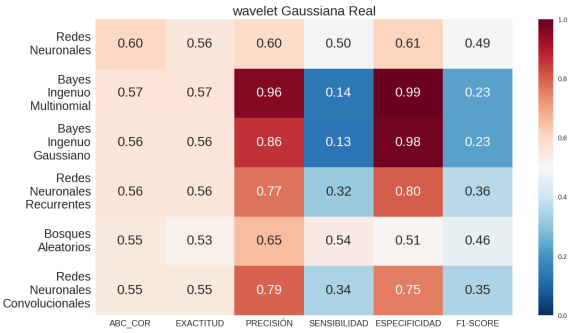
Figura 6.5: Resultados con la wavelet Gaussiana Compleja

Gaussiana Compleja

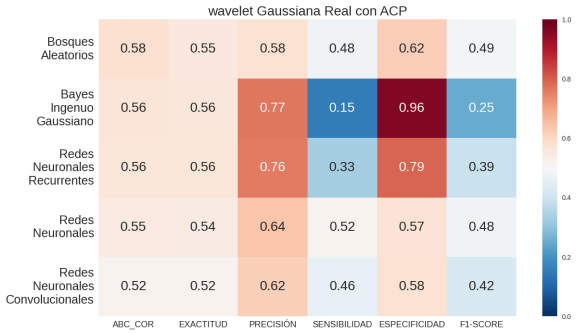
Los resultados con los datos originales (Figura 6.5a) siguen el mismo comportamiento de reconocer a la mayoría de pacientes como sanos y clasificar correctamente a la mayoría de los pocos enfermos que detecta, con diferentes grados de equilibrio entre métricas. F1-score más baja en los modelos con más desequilibrio.

Cuando ACP es usado (Figura 6.5b), la exactitud y ABC_COR se reducen un poco para todos los modelos. Bosques Aleatorios y Redes Neuronales equilibran sus métricas y aumentan su f1-score. El resto de los modelos ganan un poco de equilibrio y cambia un poco el valor de su f1-score.

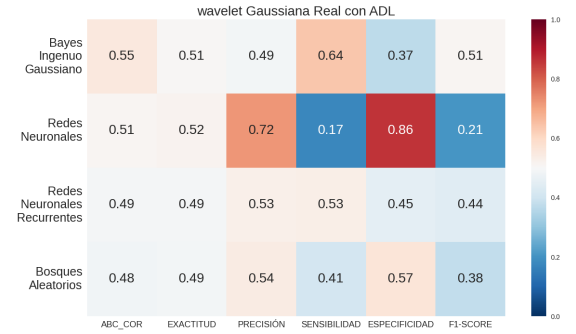
Todos los algoritmos con ADL (Figura 6.5c) equilibran sus métricas, teniendo un peor desempeño para clasificar correctamente las observaciones. Bayes Ingenuo Gaussiano es el menos perjudicado de los modelos.



(a) Variables originales



(b) ACP



(c) ADL

Figura 6.6: Resultados con la wavelet Gaussiana Real

Gaussiana Real

Los datos originales de la wavelet Gaussiana Real (Figura 6.6a) producen el modelo con el valor de ABC_COR más alto de todas las wavelets y tipos de extracción de características, las Redes Neuronales. Sus métricas presentan cierto equilibrio. Bosques Aleatorios también muestra equilibrio. El resto de los modelos tienen el comportamiento de las wavelet anteriores, con valores de precisión y especificidad altos en comparación con la sensibilidad. La f1-score se mantiene baja para todos los modelos.

Las Redes Neuronales Recurrentes y Bayes Ingenuo Gaussiano mantienen sus valores de exactitud y ABC_COR con ACP (Figura 6.6b), y el resto de sus métricas mantienen el comportamiento antes mencionado. Las Redes Neuronales empeoran en general y se desequilibran un poco. Las Redes Neuronales Convolucionales empeoran su exactitud y ABC_COR, además que se equilibran ligeramente el resto de métricas. Bosques Aleatorios tiene una mejora en exactitud y ABC_COR, a su vez que se equilibran las otras métricas.

Finalmente, con ADL (Figura 6.6c) se reducen todas las métricas. Todos los modelos ganan equilibrio excepto las Redes Neuronales.

Capítulo 7

Conclusiones

Con el desarrollo de este trabajo se logró mostrar el proceso de implementación de algoritmos de Aprendizaje Automatizado, incluyendo las etapas y consideraciones necesarias para desarrollar el Aprendizaje Automatizado. Este texto puede ser un referente para quienes quieran conocer los fundamentos de esta área de la Inteligencia Artificial, a través de un caso práctico.

En cuanto al estudio de la detección de retinopatía diabética con ERG, los resultados aquí mostrados son un punto de partida para ampliar la búsqueda de nuevos métodos enfocados a la detección de enfermedades oculares. A su vez son una referencia para el estudio de señales eléctricas biológicas o médicas, al servir como una guía de aplicación del Aprendizaje Automatizado a estas señales o como resultados para contrastar otras metodologías para procesar este tipo de señales. Los métodos de extracción de características y de optimización de hiperparámetros pueden ser replicados en conjunto al aplicar modelos de Aprendizaje Automatizado con señales similares a los ERG usados en esta obra o con señales de otro tipo de fenómenos.

Al observar el comportamiento de todos los modelos para cada wavelet y cada método de extracción de características se puede decir que en general con ACP y ADL no se incrementa la capacidad para detectar la retinopatía diabética a partir de los datos disponibles. Esto significa que las variables originales tienen información relevante que es eliminada con la extracción de características.

La disminución del puntaje de las métricas visto en ADL, puede ser provocado por tener una sola dimensión en los datos transformados que no permite hacer una separación correcta de los pacientes sanos y enfermos; Al tener sólo dos valores de variable objetivo (sano y enfermo) ADL devuelve una dimensión. Se podrían aplicar métodos de Aprendizaje no Supervisado para encontrar si los datos tienden a agruparse en tres o más categorías para aplicar ADL con más efectividad.

Ningún modelo prevalece como el peor o el mejor en todas las wavelets y tipos de extracción de características, con lo que se puede decir que hay que encontrar la wavelet adecuada para cada modelo utilizado al procesar ERG. Las Redes Neuronales con los datos originales de la wavelet Gaussiana Real (6.6a) se vuelven el mejor modelo para clasificar la retinopatía diabética

con ERG, al tener un ABC_COR de 0.60 y exactitud de 0.56. El peor modelo son las Redes Neuronales con la wavelet Gaussiana Compleja y ADL, teniendo un ABC_COR de 0.47 y una exactitud de 0.48. ACP y ADL se comportan mejor para la wavelet Sombrero Mexicano. En estudios posteriores sería relevante implementar estructuras de Redes Neuronales más complejas o incluso aplicar Redes Neuronales Pre-entrenadas, que ya han sido usadas con otras señales biológicas [84] [85] [86].

Un desempeño aceptable de los modelos con el fin de hacerlos prácticos en el diagnóstico de la retinopatía diabética implica tener más de un $80\,\%$ de exactitud o ABC_COR, por lo que los resultados promediados de los modelos hacen que no sean viables. Es posible que estos resultados sean provocados por la poca cantidad de observaciones disponibles.

Como trabajo futuro se puede replicar este estudio con mayor cantidad de datos, con el fin de verificar si las métricas mejoran y los comportamientos observados al extraer características se mantienen. Para esto es necesario obtener miles de observaciones de pacientes, lo que puede ser difícil en la práctica. Sin embargo, existen técnicas para formar miles de *datos sintéticos* a partir de un conjunto pequeño de datos originales. Estos datos nuevos mantienen las características y distribución de los datos con los que fueron creados, por lo que podrían potenciar la efectividad del entrenamiento de un modelo de Aprendizaje Automatizado [87] [88] [89].

ACP disminuye la efectividad de los modelos pero reduce el tamaño de las observaciones. Casi la totalidad de los conjuntos fueron reducidos de 71 variables objetivo a 2 o 3, representando una compactación de los datos entre un $2\,\%$ y $4\,\%$ del tamaño original. Esto puede ser una ventaja si se ejecutan los modelos con una cantidad de miles o decenas de miles de datos y se producen valores de métricas aceptables, ahorrando espacio de memoria y agilizando el procesamiento de los modelos.

En cuanto a la utilización de wavelets, Shannon y Sombrero Mexicano tienen los peores resultados y la Gaussiana Real tiene los mejores resultados. El resto de las wavelets tienen valores cercanos a la Gaussiana Real.

Al observar los resultados de los modelos con cada uno de los diez conjuntos (*Véase Apéndice A*), se aprecia que los valores de las métricas tienen variabilidad, lo que significa que los resultados de los modelos son sensibles al conjunto de datos utilizado. Esto indica la necesidad de tener más datos para probar efectivamente la capacidad de los algoritmos de Aprendizaje Automatizado utilizados.

La variabilidad en los resultados de cada conjunto también revela que los hiperparámetros obtenidos en un conjunto no necesariamente son los mejores para los demás. En estudios posteriores con más observaciones disponibles sería recomendable verificar si los hiperparámetros resultantes de la optimización se vuelven los mejores para cualquier conjunto, lo que volvería a un modelo más apto para clasificar.

Los algoritmos que más conservan sus resultados de conjunto en conjunto son Bayes Ingenuo Multinomial (figuras $A.1$ y $A.2$) y Bayes Ingenuo Gaussiano con las variables originales y con PCA (Figuras $A.3$ a $A.8$). En esas gráficas se aprecia que se clasifica a la mayoría de pacientes

como sanos. Los pocos enfermos detectados se clasifican en su mayoría correctamente. La estabilidad entre los resultados de cada conjunto puede indicar que los valores de las variables predictoras mantienen una distribución de probabilidad similar.

Referencias

[1] Saeedi, Pouya, Inga Petersohn, Paraskevi Salpea, Belma Malanda, Suvi Karuranga, Nigel Unwin, Stephen Colagiuri, Leonor Guariguata, Ayesha A. Motala, Katherine Ogurtsova, Jonathan E. Shaw, Dominic Bright y Rhys Williams: *Global and regional diabetes prevalence estimates for 2019 and projections for 2030 and 2045: Results from the International Diabetes Federation Diabetes Atlas, 9th edition*. Diabetes Research and Clinical Practice, 157:107843, 2019, ISSN 0168-8227. https://www.sciencedirect.com/science/article/pii/S0168822719312306.

[2] Subramanian, Shyamala, Sashikala Mishra, Shruti Patil, Kailash Shaw y Ebrahim Aghajari: *Machine Learning Styles for Diabetic Retinopathy Detection: A Review and Bibliometric Analysis*. Big Data and Cognitive Computing, 6(4):1, 2022, ISSN 2504-2289. https://www.mdpi.com/2504-2289/6/4/154.

[3] Teo, Zhen Ling, Yih Chung Tham, Marco Yu, Miao Li Chee, Tyler Hyungtaek Rim, Ning Cheung, Mukharram M. Bikbov, Ya Xing Wang, Yating Tang, Yi Lu, Ian Y. Wong, Daniel Shu Wei Ting, Gavin Siew Wei Tan, Jost B. Jonas, Charumathi Sabanayagam, Tien Yin Wong y Ching Yu Cheng: *Global Prevalence of Diabetic Retinopathy and Projection of Burden through 2045: Systematic Review and Meta-analysis*. Ophthalmology, 128(11):1580–1591, 2021, ISSN 0161-6420. https://www.sciencedirect.com/science/article/pii/S0161642021003213.

[4] Aracena, Claudio, Fabián Villena, Felipe Arias y Jocelyn Dunstan: *Aplicaciones de aprendizaje automático en salud*. Revista Médica Clínica Las Condes, 33(6):568–575, 2022, ISSN 0716-8640. https://www.sciencedirect.com/science/article/pii/S0716864022001195.

[5] Subramanian, Shyamala, Sashikala Mishra, Shruti Patil, Kailash Shaw y Ebrahim Aghajari: *Machine Learning Styles for Diabetic Retinopathy Detection: A Review and Bibliometric Analysis*. Big Data and Cognitive Computing, 6(4):3, 2022, ISSN 2504-2289. https://www.mdpi.com/2504-2289/6/4/154.

[6] Thébault, Stéphanie: *Pérdida de visión por diabetes: detección y tratamiento*. Revista Digital Universitaria, 20(3):1–13, 2019. http://doi.org/10.22201/codeic.16076079e.2019.v20n3.a1.

[7] Das, Dolly, Saroj Kr Biswas y Sivaji Bandyopadhyay: *A critical review on diagnosis of diabetic retinopathy using machine learning and deep learning*. Multimedia Tools and Applications, 81(18):25616–25621, 2022. https://doi.org/10.1007/s11042-022-12642-4.

[8] Doupe, Patrick, James Faghmous y Sanjay Basu: *Machine Learning for Health Services Researchers*. Value in Health, 22(7):808, 2019, ISSN 1098-3015. https://www.sciencedirect.com/science/article/pii/S1098301519301469.

[9] Bi, Qifang, Katherine E Goodman, Joshua Kaminsky y Justin Lessler: *What is Machine Learning? A Primer for the Epidemiologist*. American Journal of Epidemiology, 188(12):2222, Octubre 2019, ISSN 0002-9262. https://doi.org/10.1093/aje/kwz189.

[10] Rajula, Hema Sekhar Reddy, Giuseppe Verlato, Mirko Manchia, Nadia Antonucci y Vassilios Fanos: *Comparison of Conventional Statistical Methods with Machine Learning in Medicine: Diagnosis, Drug Development, and Treatment*. Medicina, 56(9):455, septiembre 2020, ISSN 1648-9144. http://dx.doi.org/10.3390/medicina56090455.

[11] Robson, Anthony G., Laura J. Frishman, John Grigg, Ruth Hamilton, Brett G. Jeffrey, Mineo Kondo, Shiying Li y Daphne L. McCulloch: *ISCEV Standard for full-field clinical electroretinography (2022 update)*. Documenta Ophthalmologica, 144:165–167, mayo 2022, ISSN 1573-2622. https://doi.org/10.1007/s10633-022-09872-0.

[12] Sarossy, Marc, Jonathan Crowston, Dinesh Kumar, Anne Weymouth y Zhichao Wu: *Prediction of glaucoma severity using parameters from the electroretinogram*. Scientific Reports, 11, diciembre 2021, ISSN 2045-2322. https://doi.org/10.1038/s41598-021-03421-6.

[13] Subramanian, Shyamala, Sashikala Mishra, Shruti Patil, Kailash Shaw y Ebrahim Aghajari: *Machine Learning Styles for Diabetic Retinopathy Detection: A Review and Bibliometric Analysis*. Big Data and Cognitive Computing, 6(4):10–11, 2022, ISSN 2504-2289. https://www.mdpi.com/2504-2289/6/4/154.

[14] Selvachandran, Ganeshsree, Shio Gai Quek, Raveendran Paramesran, Weiping Ding y Le Hoang Son: *Developments in the detection of diabetic retinopathy: a state-of-the-art review of computer-aided diagnosis and machine learning methods*. Artificial intelligence review, 56(2):920, 2023, ISSN 1573-7462. https://doi.org/10.1007/s10462-022-10185-6.

[15] Zhdanov, Aleksei, Anton Dolganov, Dario Zanca, Vasilii Borisov y Mikhail Ronkin: *Advanced Analysis of Electroretinograms Based on Wavelet Scalogram Processing*. Applied Sciences, 12(23):2, 2022, ISSN 2076-3417. https://www.mdpi.com/2076-3417/12/23/12365.

[16] Gajendran, Mohan Kumar, Landon J. Rohowetz, Peter Koulen y Amirfarhang Mehdizadeh: *Novel Machine-Learning Based Framework Using Electroretinography Data for the Detection of Early-Stage Glaucoma*. Frontiers in Neuroscience, 16, 2022, ISSN 1662-453X. https://www.frontiersin.org/articles/10.3389/fnins.2022.869137.

[17] Glinton, Sophie L., Antonio Calcagni, Watjana Lilaonitkul, Nikolas Pontikos, Sandra Vermeirsch, Gongyu Zhang, Gavin Arno, Siegfried K. Wagner, Michel Michaelides, Pearse A. Keane, Andrew R. Webster, Omar A. Mahroo y Anthony G. Robson: *Phenotyping of ABCA4 Retinopathy by Machine Learning Analysis of Full-Field Electroretinography*. Translational Vision Science & Technology, 11(9), Septiembre 2022, ISSN 2164-2591. https://doi.org/10.1167/tvst.11.9.34.

[18] Diao, Tina, Fareshta Kushzad, Megh D. Patel, Megha P. Bindiganavale, Munam Wasi, Mykel J. Kochenderfer y Heather E. Moss: *Comparison of Machine Learning Approaches to Improve Diagnosis of Optic Neuropathy Using Photopic Negative Response Measured Using a Handheld Device*. Frontiers in Medicine, 8, 2021, ISSN 2296-858X. https://www.frontiersin.org/articles/10.3389/fmed.2021.771713.

[19] Zhdanov, Aleksei, Anton Dolganov, Dario Zanca, Vasilii Borisov y Mikhail Ronkin: *Advanced Analysis of Electroretinograms Based on Wavelet Scalogram Processing*. Applied Sciences, 12(23), 2022, ISSN 2076-3417. https://www.mdpi.com/2076-3417/12/23/12365.

[20] Sudha, S, A Srinivasan y T Gayathri Devi: *Detection and Classification of Diabetic Retinopathy Using DCNN and BSN Models.* Computers, Materials & Continua, 72(1), 2022.

[21] Noguez Imm, Ramsés, Julio Muñoz-Benitez, Diego Medina, Everardo Barcenas, Guillermo Molero-Castillo, Pamela Reyes-Ortega, Jorge Armando Hughes-Cano, Leticia Medrano-Gracia, Manuel Miranda-Anaya, Gerardo Rojas-Piloni *y cols.*: *Preventable risk factors for type 2 diabetes can be detected using noninvasive spontaneous electroretinogram signals*. Plos one, 18(1):e0278388, 2023.

[22] Ba-Ali, Shakoor, Michael Larsen, Henrik Ullits Andersen y Henrik Lund-Andersen: *Full-field and multifocal electroretinogram in non-diabetic controls and diabetics with and without retinopathy*. Acta Ophthalmologica, 100(8):e1719–e1728, 2022. https://onlinelibrary.wiley.com/doi/abs/10.1111/aos.15184.

[23] Raschka, Sebastian y Vahid Mirjalili: *Python Machine Learning*, capítulo 1, páginas 1–5. Packt Publishing, UK, Birmingham, 3ª edición, diciembre 2019.

[24] Brownlee, Jason: *Data Preparation for Machine Learning*, capítulo 1, página 3. Jason Brownlee, 1ª edición, 2020.

[25] Raschka, Sebastian y Vahid Mirjalili: *Python Machine Learning*. Packt Publishing, 3ª edición, 2019.

[26] Janiesch, Christian, Patrick Zschech y Kai Heinrich: *Machine learning and deep learning*. Electronic Markets, 31:687, septiembre 2021, ISSN 1422-8890. https://doi.org/10.1007/s12525-021-00475-2.

[27] Bi, Qifang, Katherine E Goodman, Joshua Kaminsky y Justin Lessler: *What is Machine Learning? A Primer for the Epidemiologist*. American Journal of Epidemiology, 188(12):2223, Octubre 2019, ISSN 0002-9262. https://doi.org/10.1093/aje/kwz189.

[28] Raschka, Sebastian y Vahid Mirjalili: *Python Machine Learning*, capítulo 1, páginas 3–4. Packt Publishing, UK, Birmingham, 3ª edición, diciembre 2019.

[29] Rebala, Gopinath, Ajay Ravi y Sanjay Churiwala: *An Introduction to Machine Learning*, capítulo 2, páginas 19–22. Springer, Switzerland, 1ª edición, 2019.

[30] Raschka, Sebastian y Vahid Mirjalili: *Python Machine Learning*, capítulo 1, páginas 7–8. Packt Publishing, UK, Birmingham, 3ª edición, diciembre 2019.

[31] Rebala, Gopinath, Ajay Ravi y Sanjay Churiwala: *An Introduction to Machine Learning*, capítulo 2, página 22. Springer, Switzerland, 1ª edición, 2019.

[32] Raschka, Sebastian y Vahid Mirjalili: *Python Machine Learning*, capítulo 1, páginas 6–7. Packt Publishing, UK, Birmingham, 3ª edición, diciembre 2019.

[33] Rebala, Gopinath, Ajay Ravi y Sanjay Churiwala: *An Introduction to Machine Learning*, capítulo 16, páginas 195–198. Springer, Switzerland, 1ª edición, 2019.

[34] Raschka, Sebastian y Vahid Mirjalili: *Python Machine Learning*, capítulo 1, páginas 12–14. Packt Publishing, UK, Birmingham, 3ª edición, diciembre 2019.

[35] Janiesch, Christian, Patrick Zschech y Kai Heinrich: *Machine learning and deep learning*. Electronic Markets, 31:688–691, septiembre 2021, ISSN 1422-8890. https://doi.org/10.1007/s12525-021-00475-2.

[36] Janiesch, Christian, Patrick Zschech y Kai Heinrich: *Machine learning and deep learning*. Electronic Markets, 31:688, septiembre 2021, ISSN 1422-8890. https://doi.org/10.1007/s12525-021-00475-2.

[37] Janiesch, Christian, Patrick Zschech y Kai Heinrich: *Machine learning and deep learning*. Electronic Markets, 31:691, septiembre 2021, ISSN 1422-8890. https://doi.org/10.1007/s12525-021-00475-2.

[38] Zebari, Rizgar, Adnan Abdulazeez, Diyar Zeebaree, Dilovan Zebari y Jwan Saeed: *A Comprehensive Review of Dimensionality Reduction Techniques for Feature Selection and Feature Extraction*. Journal of Applied Science and Technology Trends, 1(2):56 – 70, Mayo 2020. https://www.jastt.org/index.php/jasttpath/article/view/24.

[39] Raschka, Sebastian y Vahid Mirjalili: *Python Machine Learning*, capítulo 5, página 145. Packt Publishing, UK, Birmingham, 3ª edición, diciembre 2019.

[40] Raschka, Sebastian y Vahid Mirjalili: *Python Machine Learning*, capítulo 1, página 13. Packt Publishing, UK, Birmingham, 3ª edición, diciembre 2019.

[41] Raschka, Sebastian y Vahid Mirjalili: *Python Machine Learning*, capítulo 1, página 53. Packt Publishing, UK, Birmingham, 3ª edición, diciembre 2019.

[42] Janiesch, Christian, Patrick Zschech y Kai Heinrich: *Machine learning and deep learning*. Electronic Markets, 31:689, septiembre 2021, ISSN 1422-8890. https://doi.org/10.1007/s12525-021-00475-2.

[43] Raschka, Sebastian y Vahid Mirjalili: *Python Machine Learning*, capítulo 1, páginas 12–13. Packt Publishing, UK, Birmingham, 3ª edición, diciembre 2019.

[44] Raschka, Sebastian y Vahid Mirjalili: *Python Machine Learning*, capítulo 4, páginas 121–124. Packt Publishing, UK, Birmingham, 3ª edición, diciembre 2019.

[45] Bischl, Bernd, Martin Binder, Michel Lang, Tobias Pielok, Jakob Richter, Stefan Coors, Janek Thomas, Theresa Ullmann, Marc Becker, Anne Laure Boulesteix, Difan Deng y Marius Lindauer: *Hyperparameter optimization: Foundations, algorithms, best practices, and open challenges*. WIREs Data Mining and Knowledge Discovery, 13(2):e1484, 2023. https://wires.onlinelibrary.wiley.com/doi/abs/10.1002/widm.1484.

[46] Breiman, Leo: *Random Forests*. Machine Learning, 45:5–32, octubre 2001, ISSN 1573-0565. https://doi.org/10.1023/A:1010933404324.

[47] Schonlau, Matthias y Rosie Yuyan Zou: *The random forest algorithm for statistical learning*. The Stata Journal, 20(1):3–29, 2020. https://doi.org/10.1177/1536867X20909688.

[48] Aria, Massimo, Corrado Cuccurullo y Agostino Gnasso: *A comparison among interpretative proposals for Random Forests*. Machine Learning with Applications, 6:100094, 2021,

ISSN 2666-8270. https://www.sciencedirect.com/science/article/pii/S2666827021000475.

[49] Zhang, Harry: *The optimality of naive Bayes*. Aa, 1(2):3, 2004.

[50] Damanik, F J y D B Setyohadi: *Analysis Of Public Sentiment About Covid-19 In Indonesia On Twitter Using Multinomial Naive Bayes And Support Vector Machine*. IOP Conference Series: Earth and Environmental Science, 704(1):012027, Marzo 2021. https://dx.doi.org/10.1088/1755-1315/704/1/012027.

[51] Raschka, Sebastian y Vahid Mirjalili: *Python Machine Learning*, capítulo 2, páginas 20–26. Packt Publishing, UK, Birmingham, 3ª edición, diciembre 2019.

[52] Staudemeyer, Ralf C y Eric Rothstein Morris: *Understanding LSTM–a tutorial into long short-term memory recurrent neural networks*. arXiv preprint arXiv:1909.09586, páginas 3–4, 2019.

[53] Staudemeyer, Ralf C y Eric Rothstein Morris: *Understanding LSTM–a tutorial into long short-term memory recurrent neural networks*. arXiv preprint arXiv:1909.09586, páginas 4–5, 2019.

[54] Staudemeyer, Ralf C y Eric Rothstein Morris: *Understanding LSTM–a tutorial into long short-term memory recurrent neural networks*. arXiv preprint arXiv:1909.09586, páginas 7–11, 2019.

[55] Goodfellow, Ian, Yoshua Bengio y Aaron Courville: *Deep Learning*, capítulo 4, páginas 80–91. MIT Press, 2016. http://www.deeplearningbook.org.

[56] Goodfellow, Ian, Yoshua Bengio y Aaron Courville: *Deep Learning*, capítulo 5, páginas 149–150. MIT Press, 2016. http://www.deeplearningbook.org.

[57] Goodfellow, Ian, Yoshua Bengio y Aaron Courville: *Deep Learning*, capítulo 8, páginas 274–279. MIT Press, 2016. http://www.deeplearningbook.org.

[58] Arana, Carlos: *Redes neuronales recurrentes: Análisis de los modelos especializados en datos secuenciales*. Informe técnico, Serie Documentos de Trabajo, 2021.

[59] Rebala, Gopinath, Ajay Ravi y Sanjay Churiwala: *An Introduction to Machine Learning*, capítulo 11, páginas 127–134. Springer, Switzerland, 1ª edición, 2019.

[60] Goodfellow, Ian, Yoshua Bengio y Aaron Courville: *Deep Learning*, capítulo 10, páginas 368–390. MIT Press, 2016. http://www.deeplearningbook.org.

[61] Goodfellow, Ian, Yoshua Bengio y Aaron Courville: *Deep Learning*, capítulo 9, páginas 326–366. MIT Press, 2016. http://www.deeplearningbook.org.

[62] García Aquino, Christian, Dante Mújica-Vargas, Gabriel Serna y Manuel De Jesús Cruz: *Implementation of Deep Neural Network Models on Embedded GPUs for Classification of Atrial Fibrillation Sequences*. Research in Computing Science, 151:129–142, Mayo 2022.

[63] Rebala, Gopinath, Ajay Ravi y Sanjay Churiwala: *An Introduction to Machine Learning*, capítulo 15, páginas 181–194. Springer, Switzerland, 1ª edición, 2019.

[64] Zebari, Rizgar, Adnan Abdulazeez, Diyar Zeebaree, Dilovan Zebari y Jwan Saeed: *A Comprehensive Review of Dimensionality Reduction Techniques for Feature Selection and Feature Extraction*. Journal of Applied Science and Technology Trends, 1(2):59, Mayo 2020. https://www.jastt.org/index.php/jasttpath/article/view/24.

[65] Balakrishnama, Suresh y Aravind Ganapathiraju: *Linear discriminant analysis-a brief tutorial*. Institute for Signal and information Processing, 18(1998):1, 1998.

[66] Lasalvia, Maria, Vito Capozzi y Giuseppe Perna: *A Comparison of PCA-LDA and PLS-DA Techniques for Classification of Vibrational Spectra*. Applied Sciences, 12(11), 2022, ISSN 2076-3417. https://www.mdpi.com/2076-3417/12/11/5345.

[67] Greenacre, Michael, Patrick Groenen, Trevor Hastie, Alfonso Iodice D'Enza, Angelos Markos y Elena Tuzhilina: *Principal component analysis*. Nature Reviews Methods Primers, 2:100, Diciembre 2022.

[68] Raschka, Sebastian y Vahid Mirjalili: *Python Machine Learning*, capítulo 5, páginas 145–159. Packt Publishing, UK, Birmingham, 3ª edición, diciembre 2019.

[69] Pernice, Sergio A: *Serie de machine learning: Análisis de Componentes Principales (PCA)*. Informe técnico, UCEMA, Serie Documentos de Trabajo, 2020.

[70] Gadekallu, Thippa Reddy, Neelu Khare, Sweta Bhattacharya, Saurabh Singh, Praveen Kumar Reddy Maddikunta, In Ho Ra y Mamoun Alazab: *Early Detection of Diabetic Retinopathy Using PCA-Firefly Based Deep Learning Model*. Electronics, 9(2):274, Febrero 2020, ISSN 2079-9292. http://dx.doi.org/10.3390/electronics9020274.

[71] Balakrishnama, Suresh y Aravind Ganapathiraju: *Linear discriminant analysis-a brief tutorial*. Institute for Signal and information Processing, 18(1998):1–8, 1998.

[72] Gourisaria, Mahendra Kumar, Gaurav Jee, G. M. Harshvardhan, Vijander Singh, Pradeep Kumar Singh y Tewabe Chekole Workneh: *Data science appositeness in diabetes mellitus diagnosis for healthcare systems of developing nations*. IET Communications, 16(5):532–547, 2022. https://ietresearch.onlinelibrary.wiley.com/doi/abs/10.1049/cmu2.12338.

[73] Raschka, Sebastian y Vahid Mirjalili: *Python Machine Learning*, capítulo 5, páginas 159–167. Packt Publishing, UK, Birmingham, 3ª edición, diciembre 2019.

[74] Raschka, Sebastian y Vahid Mirjalili: *Python Machine Learning*, capítulo 6, páginas 216–218. Packt Publishing, UK, Birmingham, 3ª edición, diciembre 2019.

[75] Feurer, Matthias y Frank Hutter: *Hyperparameter optimization*. Springer International Publishing, 2019, ISBN 978-3-030-05318-5.

[76] Akiba, Takuya, Shotaro Sano, Toshihiko Yanase, Takeru Ohta y Masanori Koyama: *Optuna: A Next-Generation Hyperparameter Optimization Framework*. En *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '19, página 2623–2631, New York, NY, USA, 2019. Association for Computing Machinery, ISBN 9781450362016. https://doi.org/10.1145/3292500.3330701.

[77] Lai, Jung Pin, Ying Lei Lin, Ho Chuan Lin, Chih Yuan Shih, Yu Po Wang y Ping Feng Pai: *Tree-Based Machine Learning Models with Optuna in Predicting Impedance Values for Circuit Analysis*. Micromachines, 14(2), 2023, ISSN 2072-666X. https://www.mdpi.com/2072-666X/14/2/265.

[78] Hutter, Frank, Lars Kotthoff y Joaquin Vanschoren: *Automated Machine Learning*, capítulo 1, páginas 9–13. Springer International Publishing, 2019, ISBN 978-3-030-05318-5.

[79] Nomura, Masahiro, Shuhei Watanabe, Youhei Akimoto, Yoshihiko Ozaki y Masaki Onishi: *Warm Starting CMA-ES for Hyperparameter Optimization*. Proceedings of the AAAI Conference on Artificial Intelligence, 35(10):9188–9196, May 2021. https://ojs.aaai.org/index.php/AAAI/article/view/17109.

[80] Behbahani, Soroor, Hamid Ahmadieh y Sreeraman Rajan: *Feature Extraction Methods for Electroretinogram Signal Analysis: A Review*. IEEE Access, 9:116879–116897, 2021.

[81] Salles, Rafael S. y Paulo F. Ribeiro: *The use of deep learning and 2-D wavelet scalograms for power quality disturbances classification*. Electric Power Systems Research, 214:108834, 2023, ISSN 0378-7796. https://www.sciencedirect.com/science/article/pii/S0378779622008872.

[82] Ahmadieh, Hamid, Soroor Behbahani y Sare Safi: *Continuous wavelet transform analysis of ERG in patients with diabetic retinopathy*. Documenta Ophthalmologica, 142:305–314, 2021.

[83] Ngui, Wai Keng, M Salman Leong, Lim Meng Hee y Ahmed M Abdelrhman: *Wavelet analysis: mother wavelet selection methods*. Applied mechanics and materials, 393:953–958, 2013.

[84] Alquran, Hiam, Ali Mohammad Alqudah, Isam Abu-Qasmieh, Alaa Al-Badarneh y Sami Almashaqbeh: *ECG classification using higher order spectral estimation and deep learning techniques*. Neural Network World, 29(4):207–219, 2019.

[85] Khalifa, Nour Eldeen M, Mohamed Loey, Mohamed Hamed N Taha y Hamed Nasr Eldin T Mohamed: *Deep transfer learning models for medical diabetic retinopathy detection*. Acta Informatica Medica, 27(5):327, 2019.

[86] Byeon, Yeong Hyeon, Sung Bum Pan y Keun Chang Kwak: *Intelligent deep models based on scalograms of electrocardiogram signals for biometrics*. Sensors, 19(4):935, 2019.

[87] Kar, Amlan, Aayush Prakash, Ming Yu Liu, Eric Cameracci, Justin Yuan, Matt Rusiniak, David Acuna, Antonio Torralba y Sanja Fidler: *Meta-sim: Learning to generate synthetic datasets*. En *Proceedings of the IEEE/CVF International Conference on Computer Vision*, páginas 4551–4560, 2019.

[88] Goncalves, Andre, Priyadip Ray, Braden Soper, Jennifer Stevens, Linda Coyle y Ana Paula Sales: *Generation and evaluation of synthetic patient data*. BMC medical research methodology, 20(1):1–40, 2020.

[89] Dahmen, Jessamyn y Diane Cook: *SynSys: A Synthetic Data Generation System for Healthcare Applications*. Sensors, 19(5), 2019, ISSN 1424-8220. https://www.mdpi.com/1424-8220/19/5/1181.

Apéndice A

Métricas obtenidas en todos los experimentos realizados

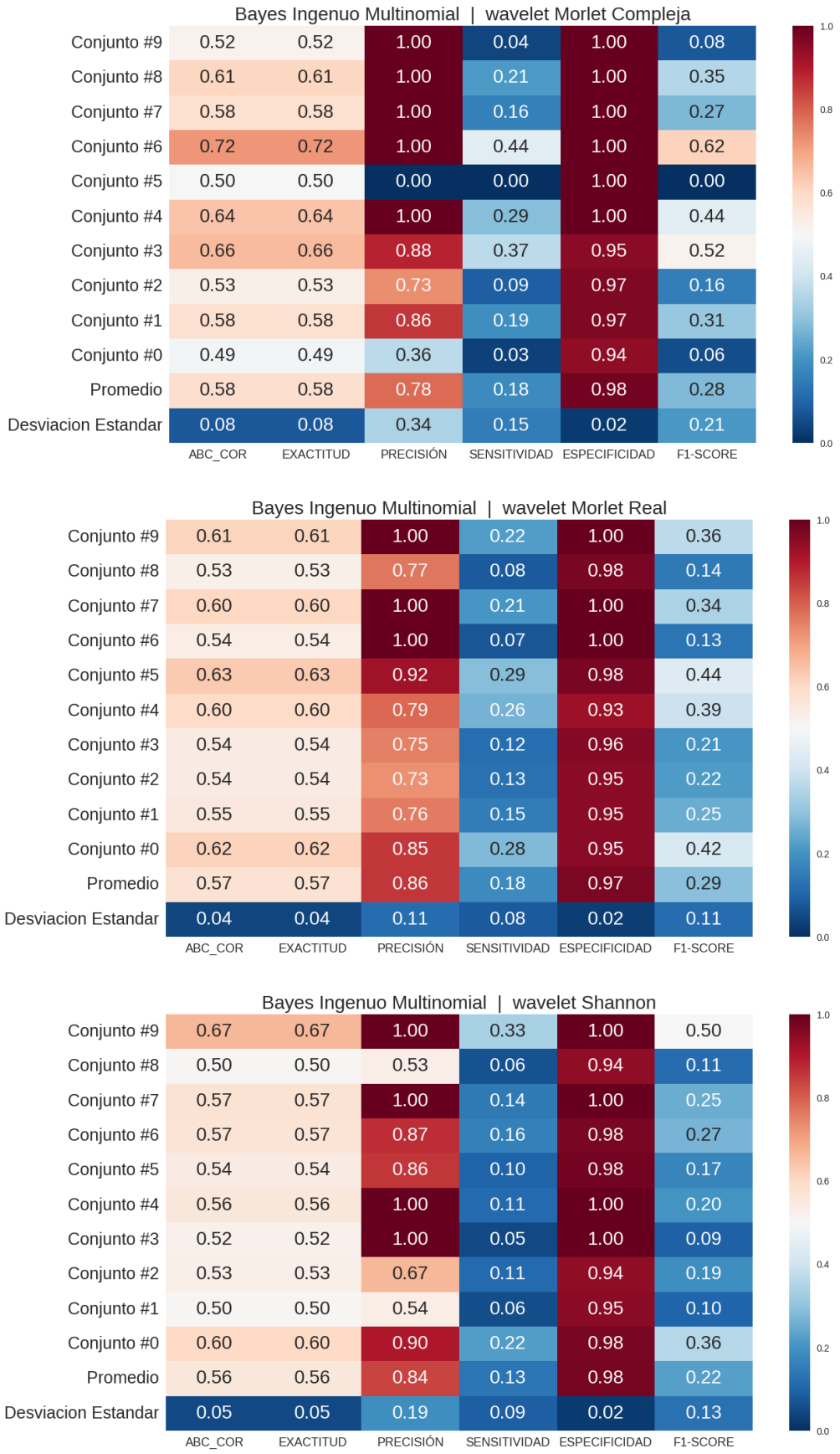


Figura A.1: Bayes Ingenuo Multinomial con las wavelets Morlet Compleja, Morlet Real y Shannon.

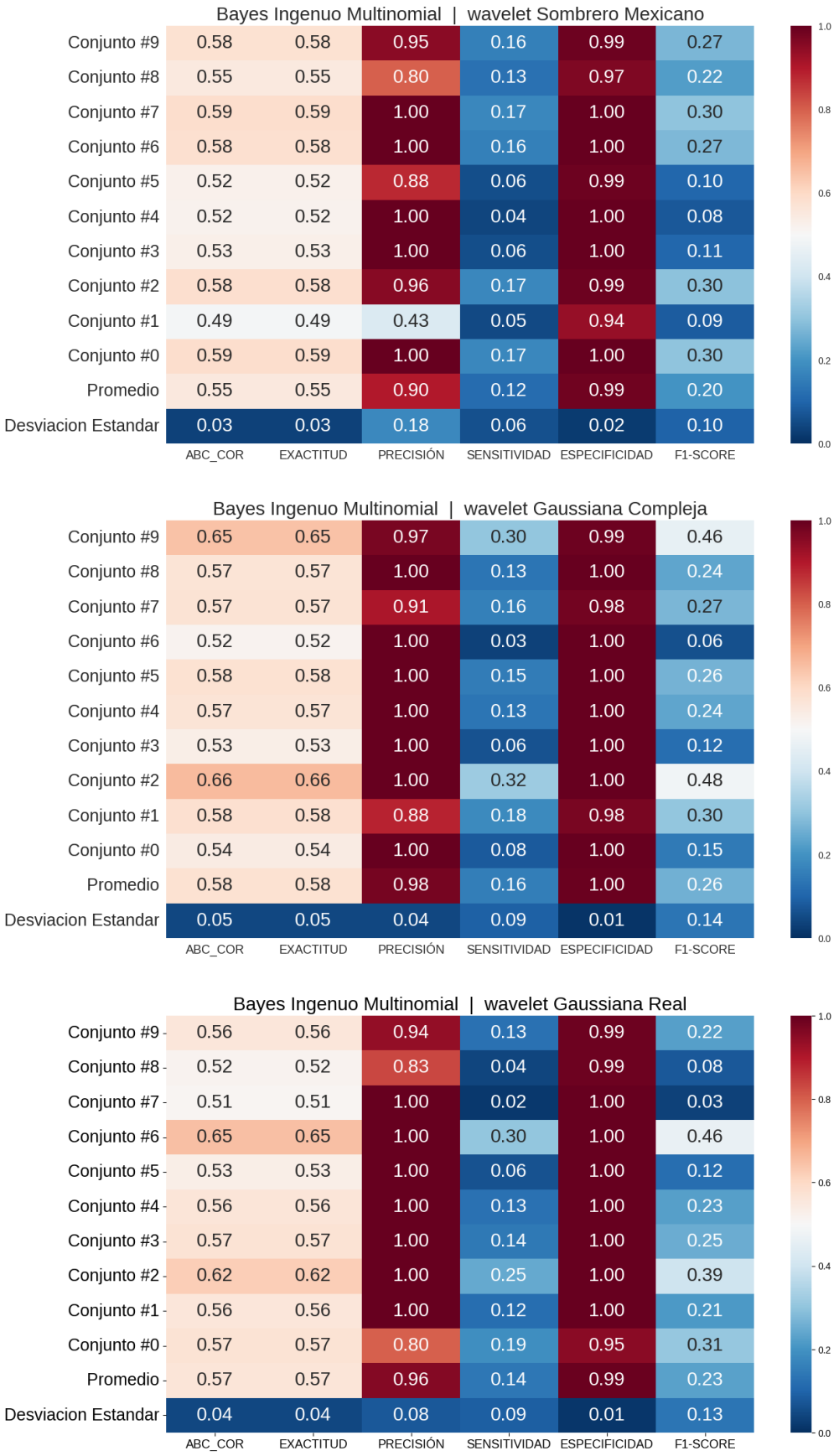


Figura A.2: Bayes Ingenuo Multinomial con las wavelets Sombrero Mexicano, Gaussiana Compleja y Gaussiana Real.

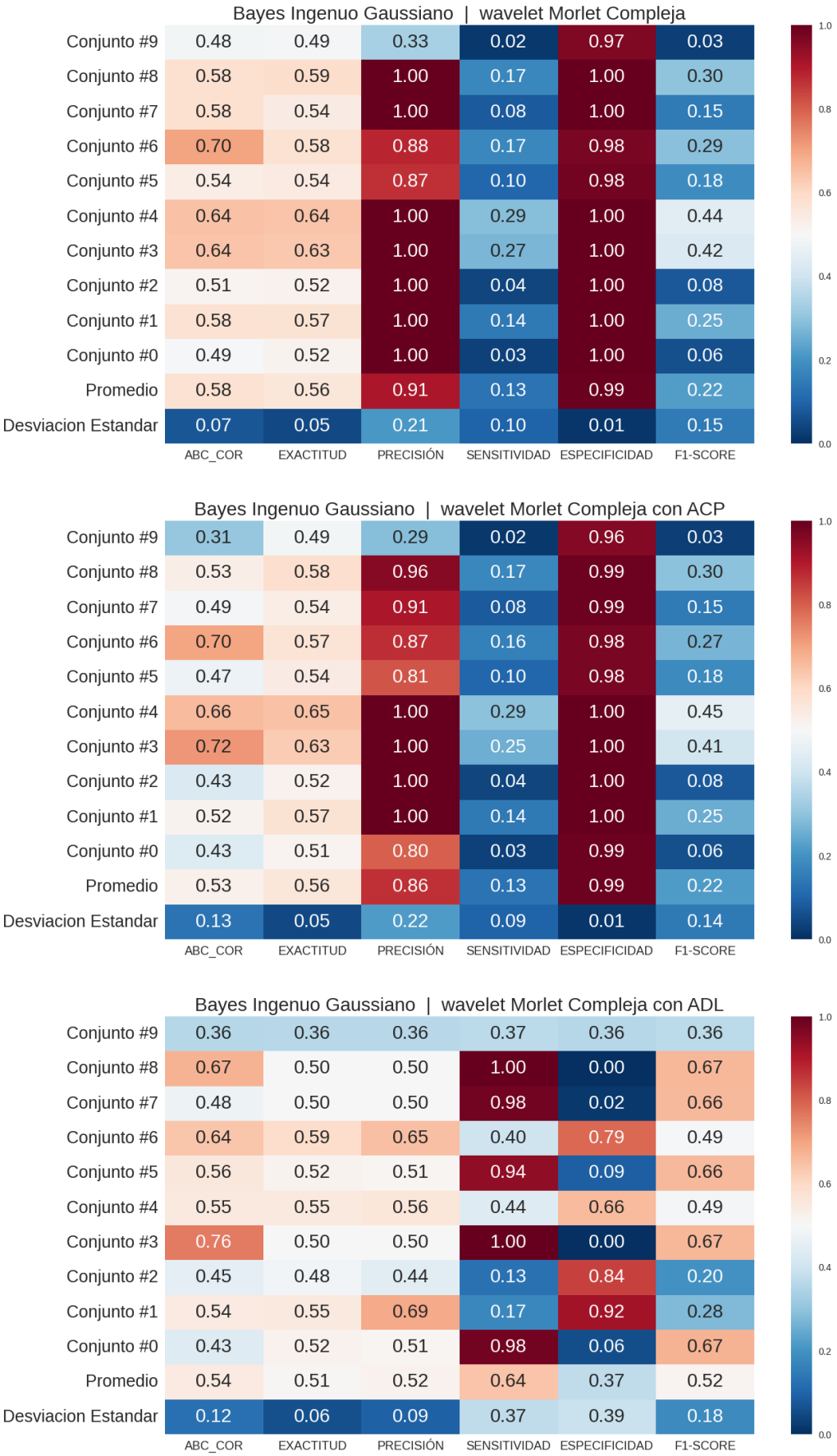


Figura A.3: Bayes Ingenuo Gaussiano con la wavelet Morlet Compleja.

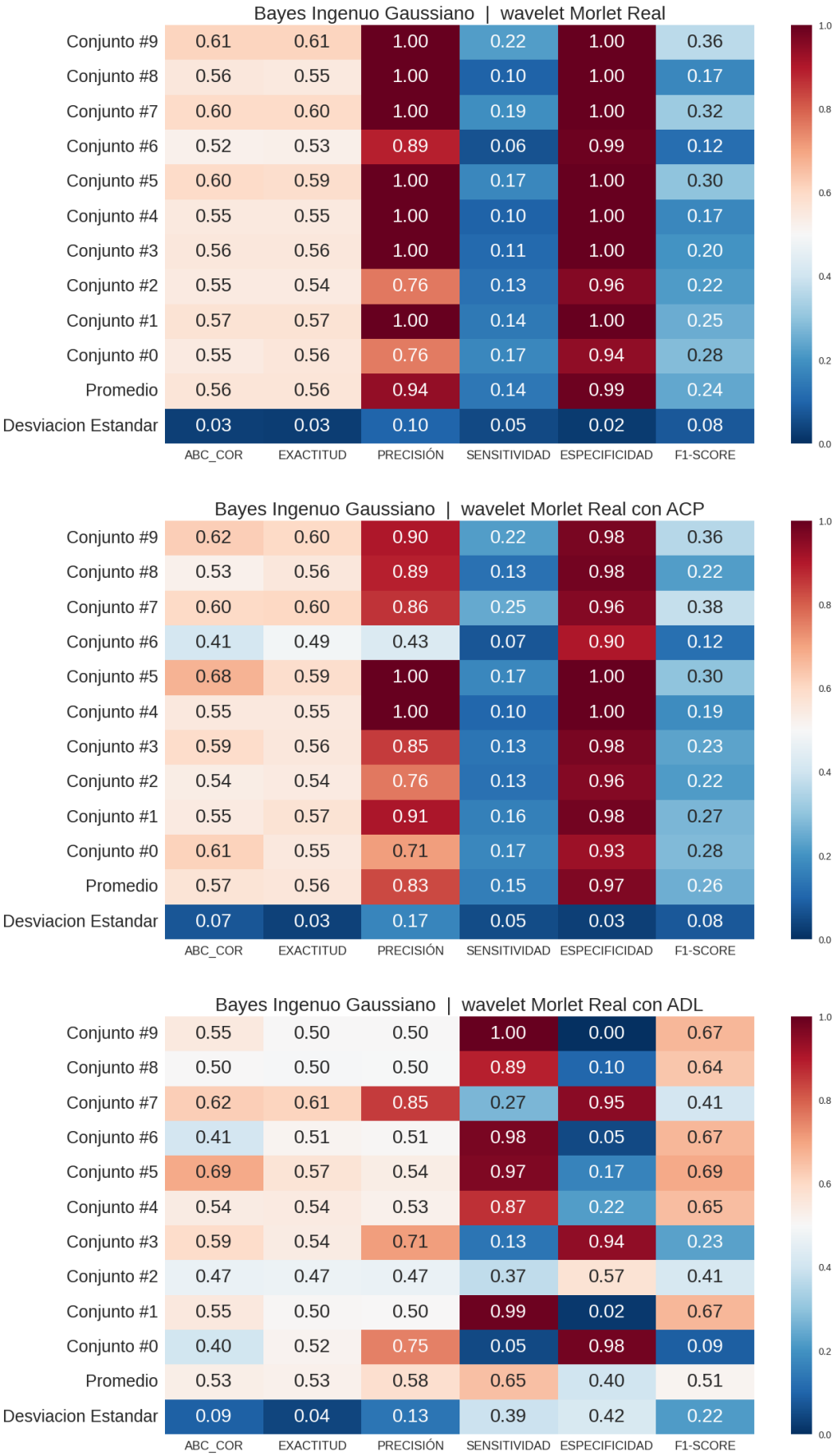


Figura A.4: Bayes Ingenuo Gaussiano con la wavelet Morlet Real.

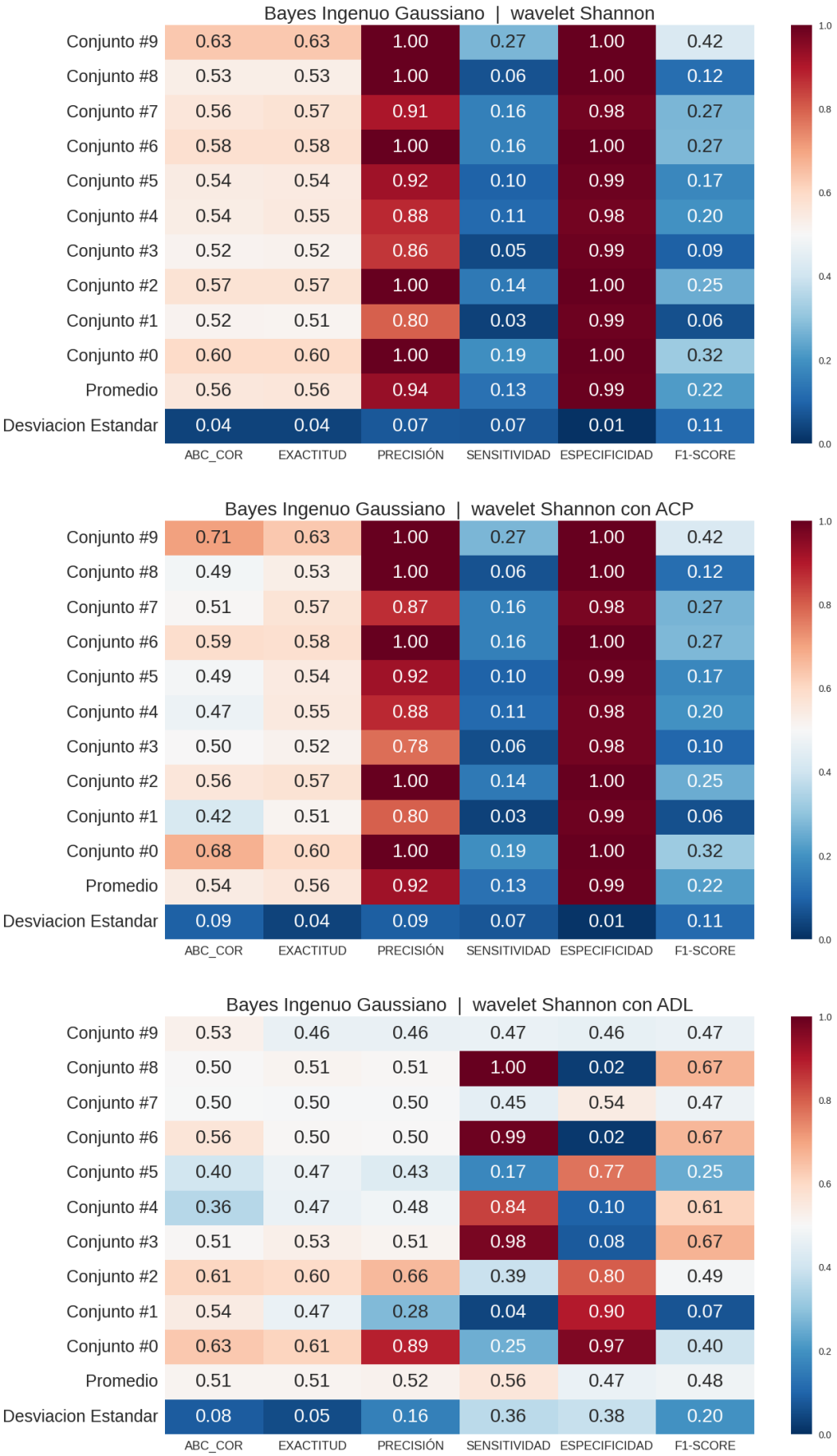


Figura A.5: Bayes Ingenuo Gaussiano con la wavelet Shannon.

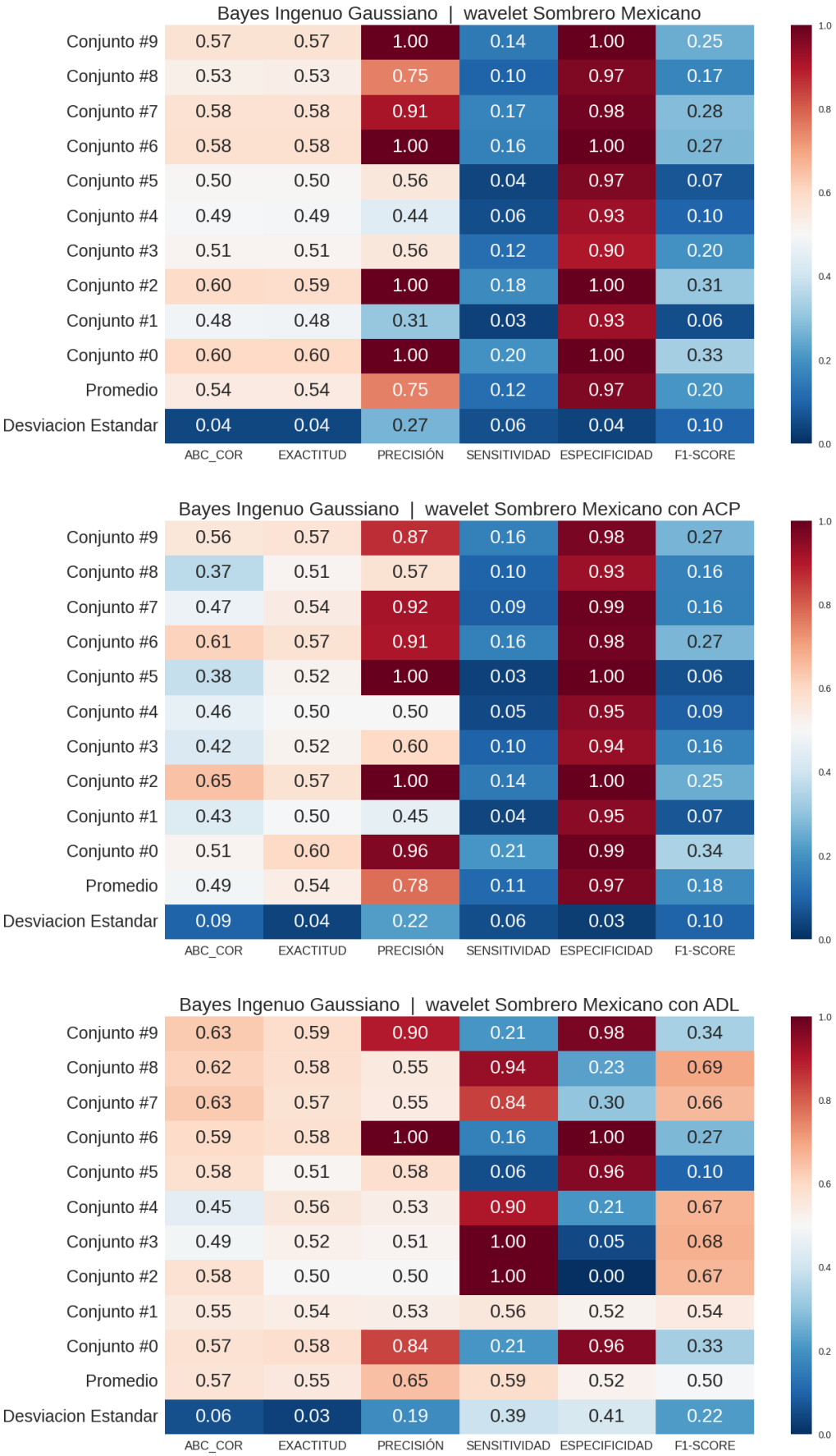


Figura A.6: Bayes Ingenuo Gaussiano con la wavelet Sombrero Mexicano.

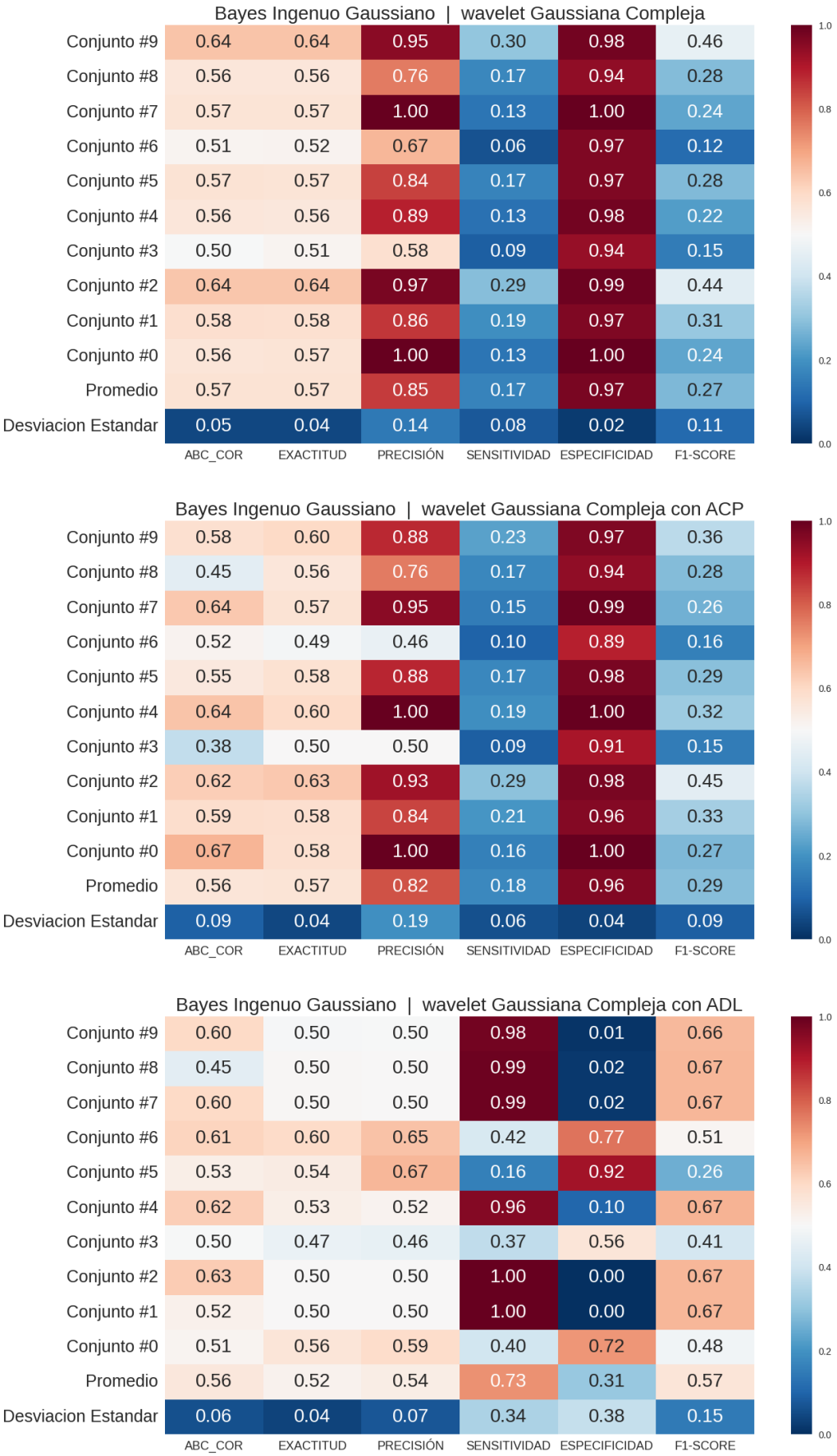


Figura A.7: Bayes Ingenuo Gaussiano con la wavelet Gaussiana Compleja.

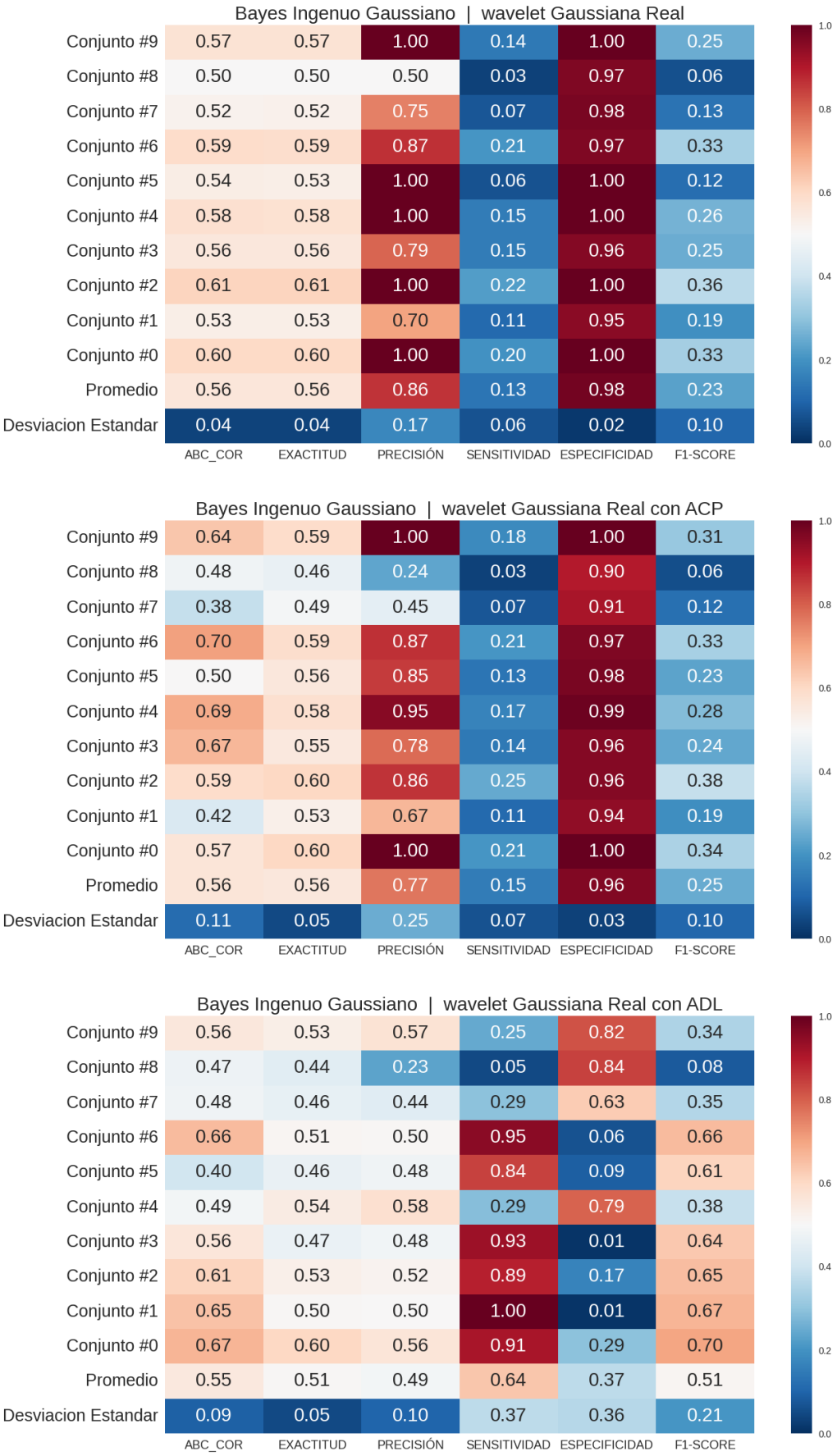


Figura A.8: Bayes Ingenuo Gaussiano con la wavelet Gaussiana Real.

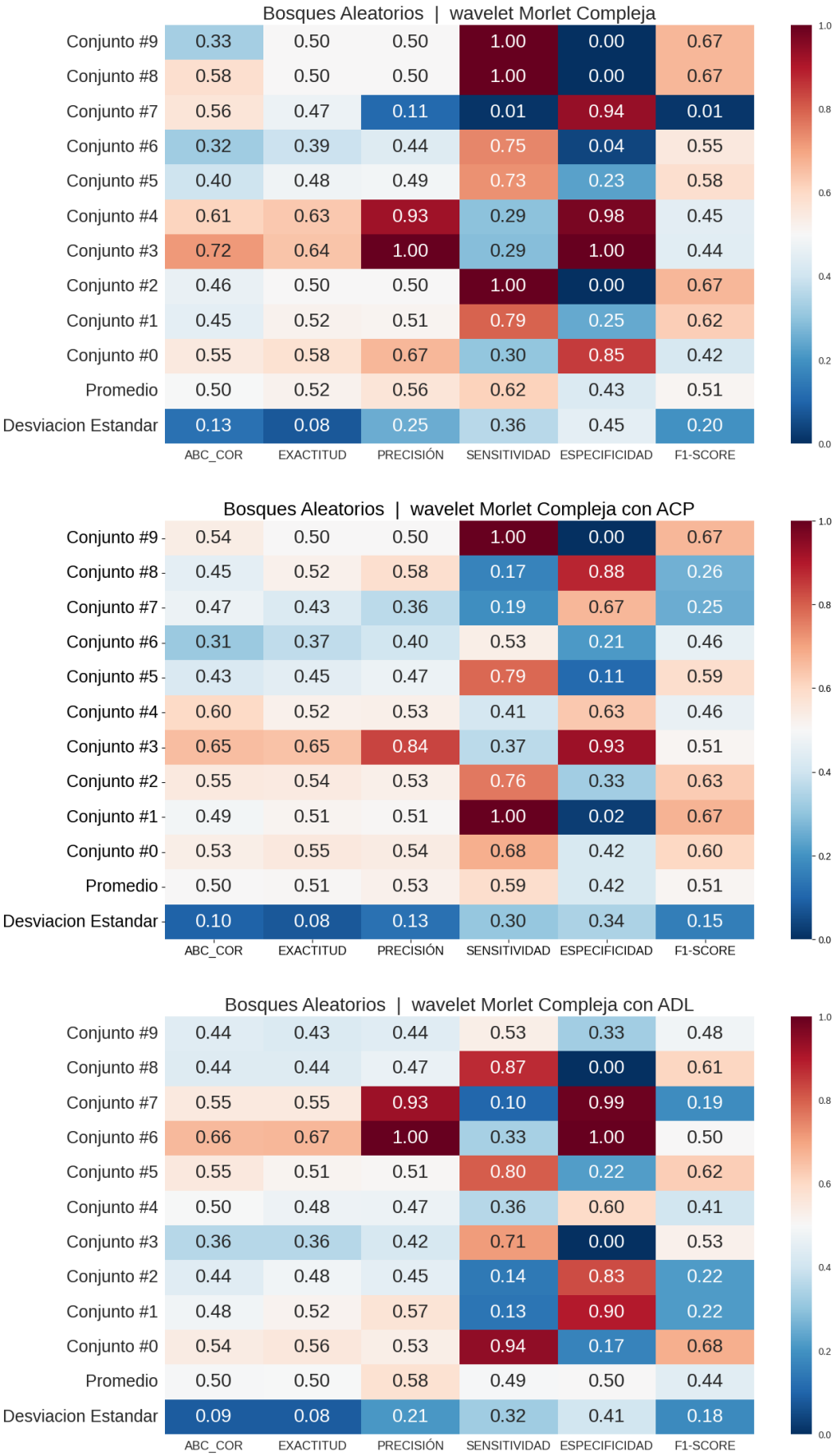


Figura A.9: Bosques Aleatorios con la wavelet Morlet Compleja.

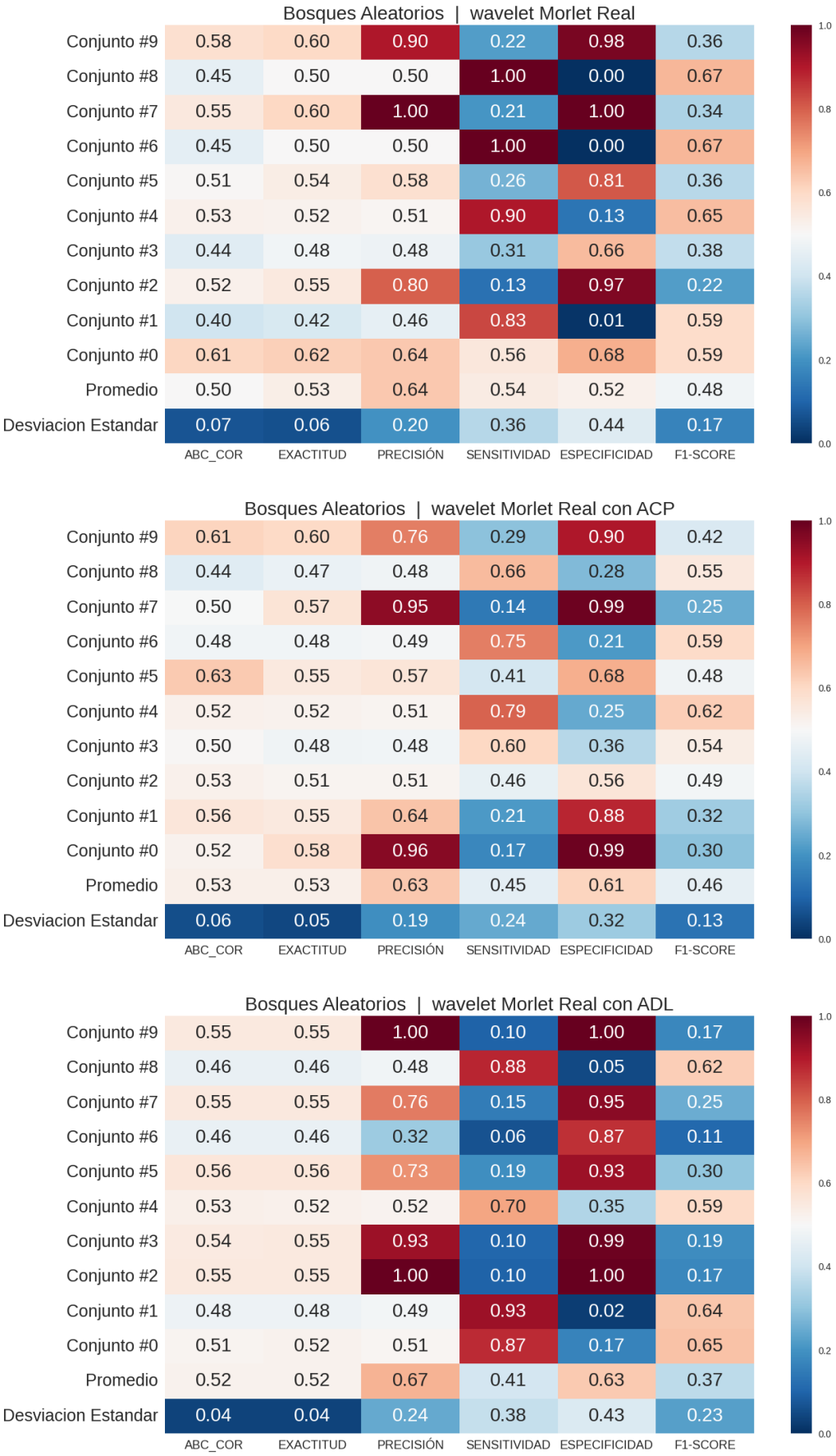


Figura A.10: Bosques Aleatorios con la wavelet Morlet Real.

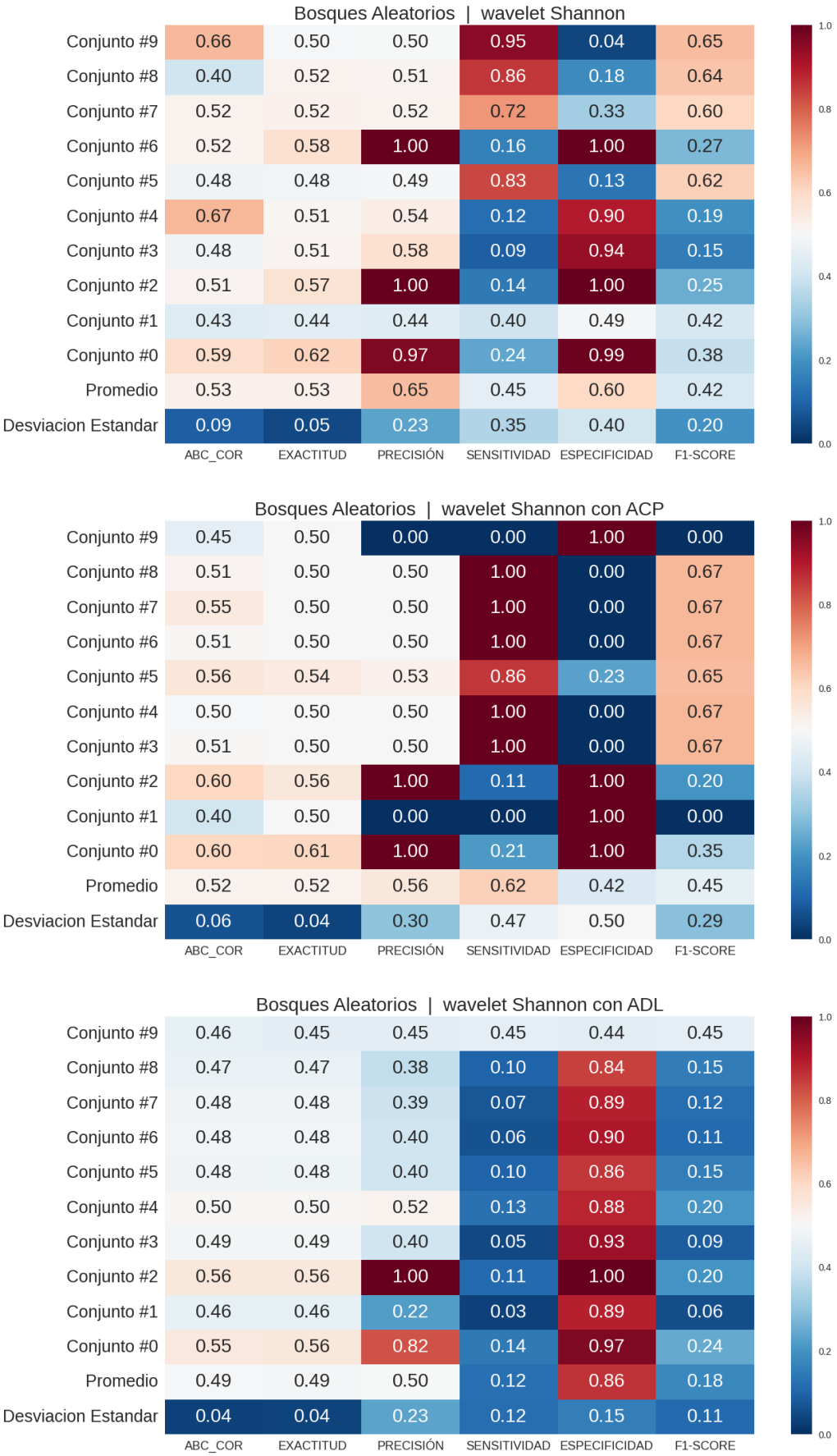


Figura A.11: Bosques Aleatorios con la wavelet Shannon.

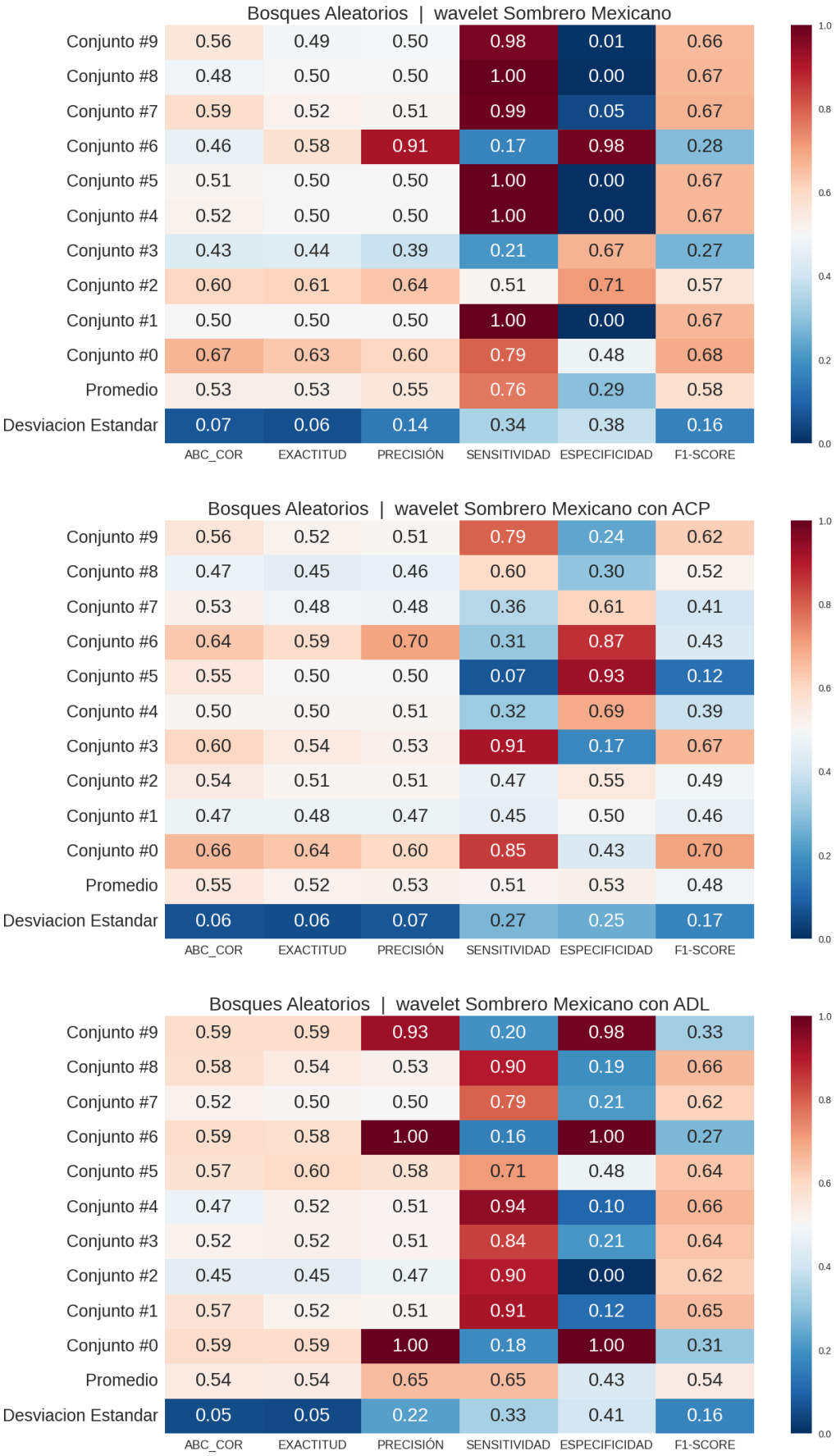


Figura A.12: Bosques Aleatorios con la wavelet Sombrero Mexicano.

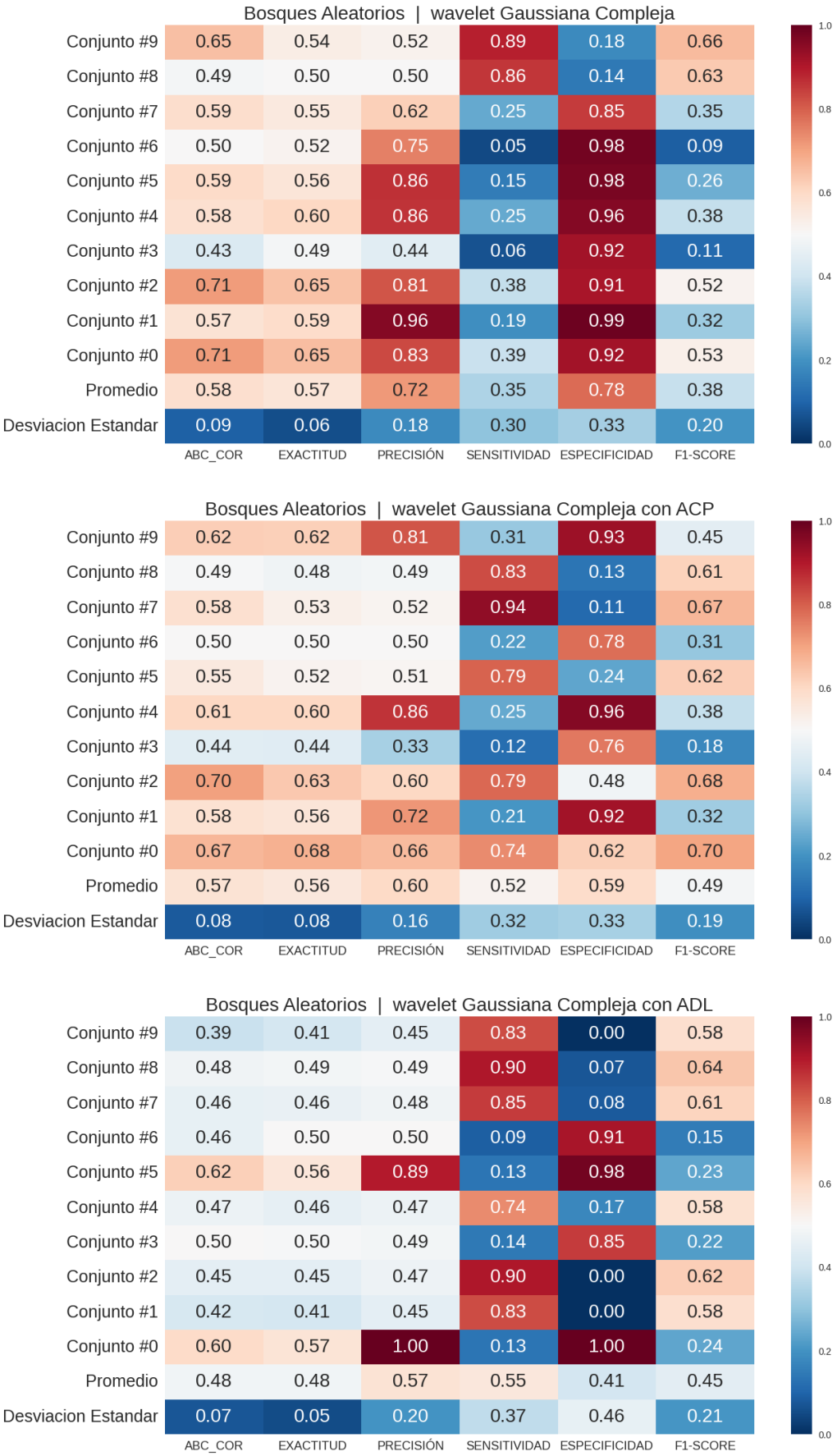


Figura A.13: Bosques Aleatorios con la wavelet Gaussiana Compleja.

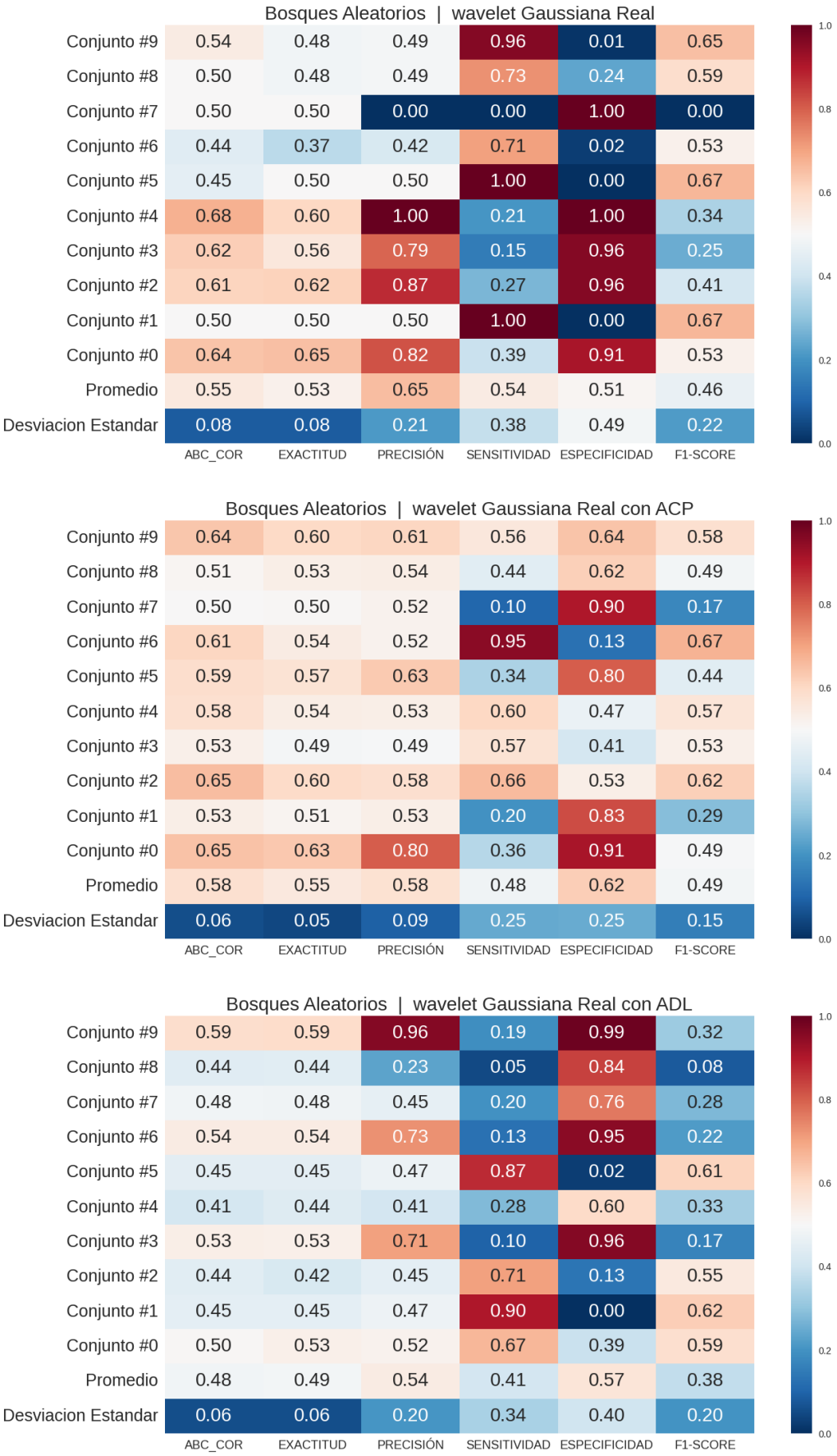


Figura A.14: Bosques Aleatorios con la wavelet Gaussiana Real.

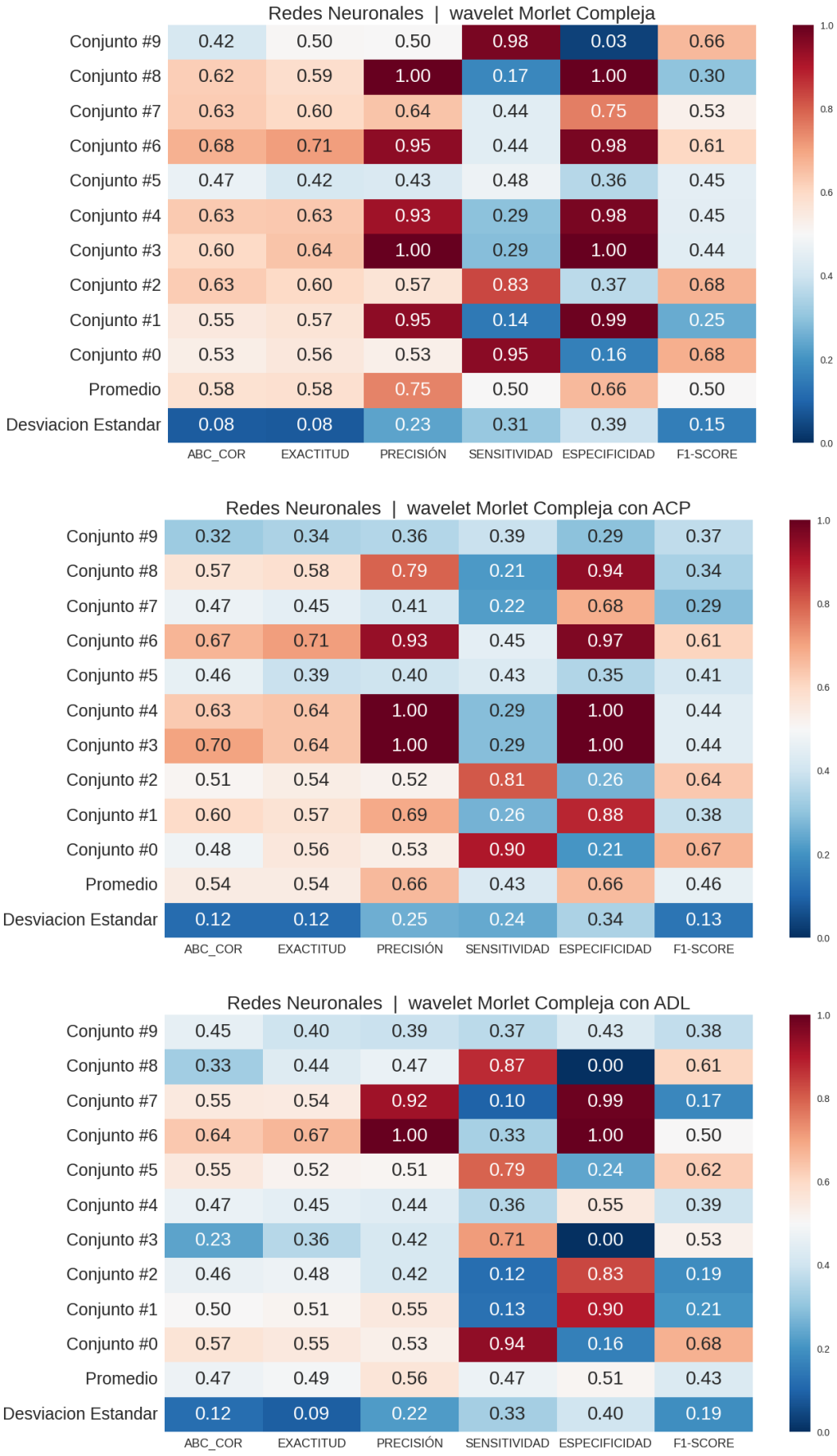


Figura A.1: Redes Neuronales con la wavelet Morlet Compleja.

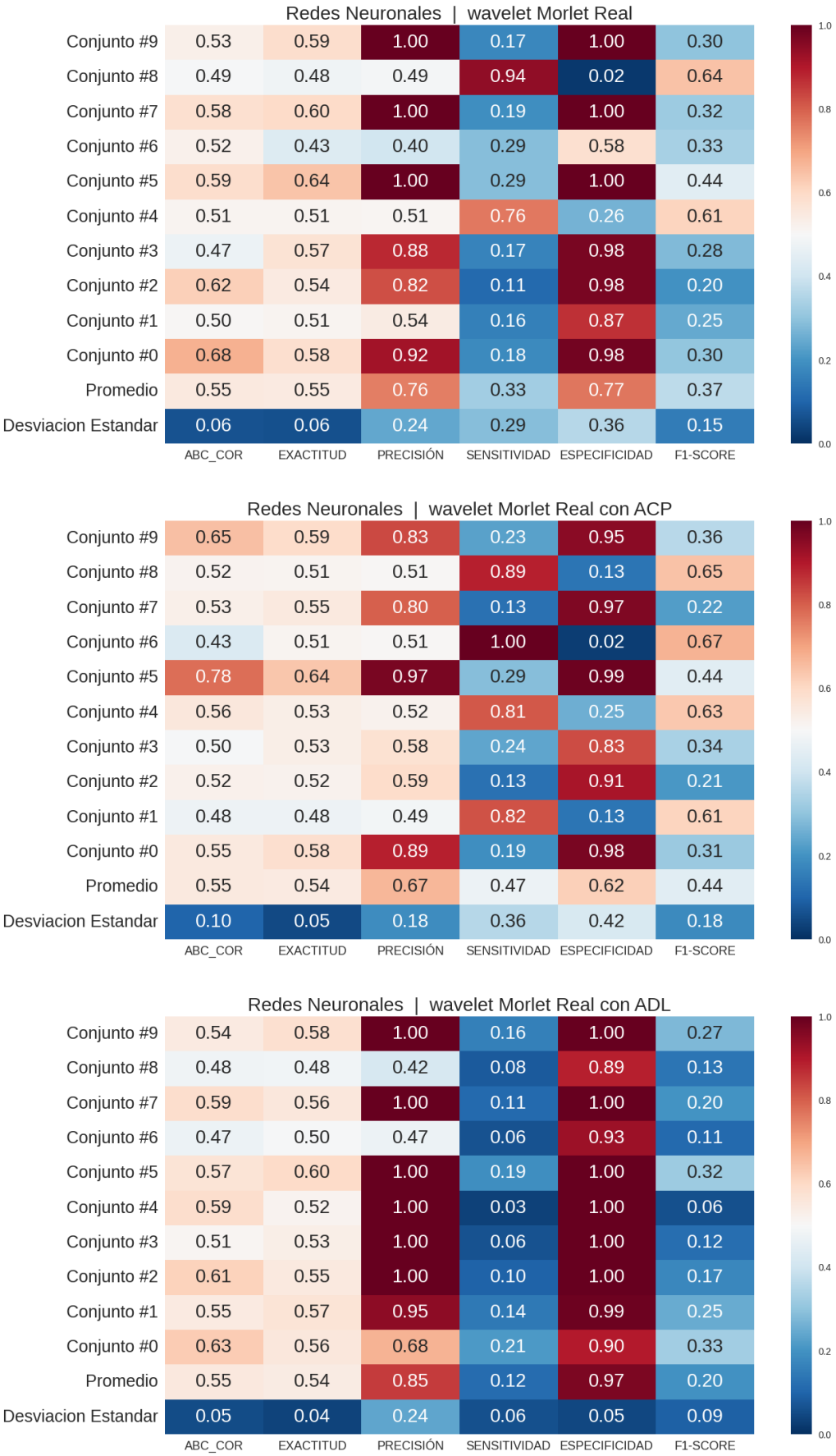


Figura A.2: Redes Neuronales con la wavelet Morlet Real.

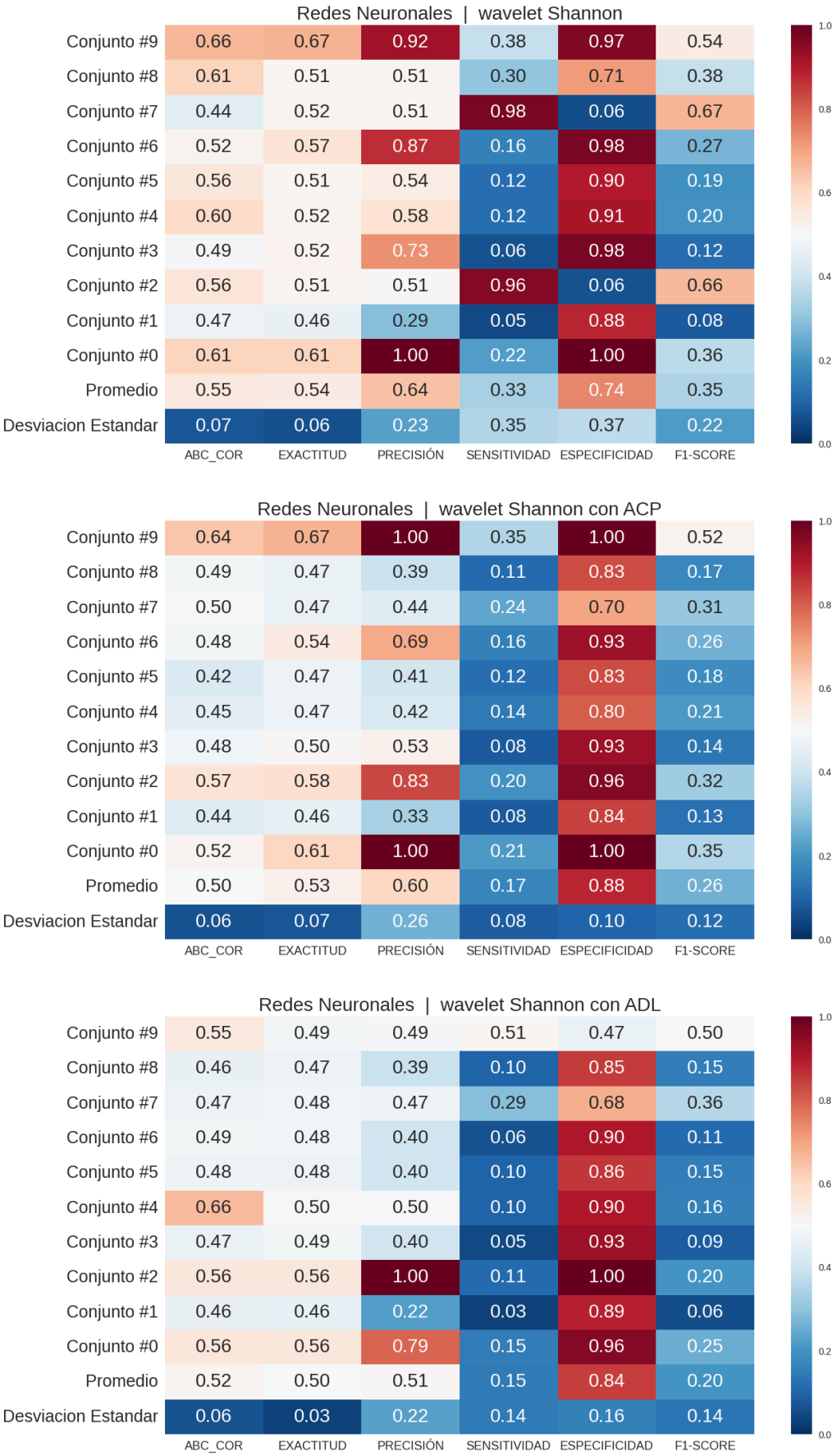


Figura A.3: Redes Neuronales con la wavelet Shannon.

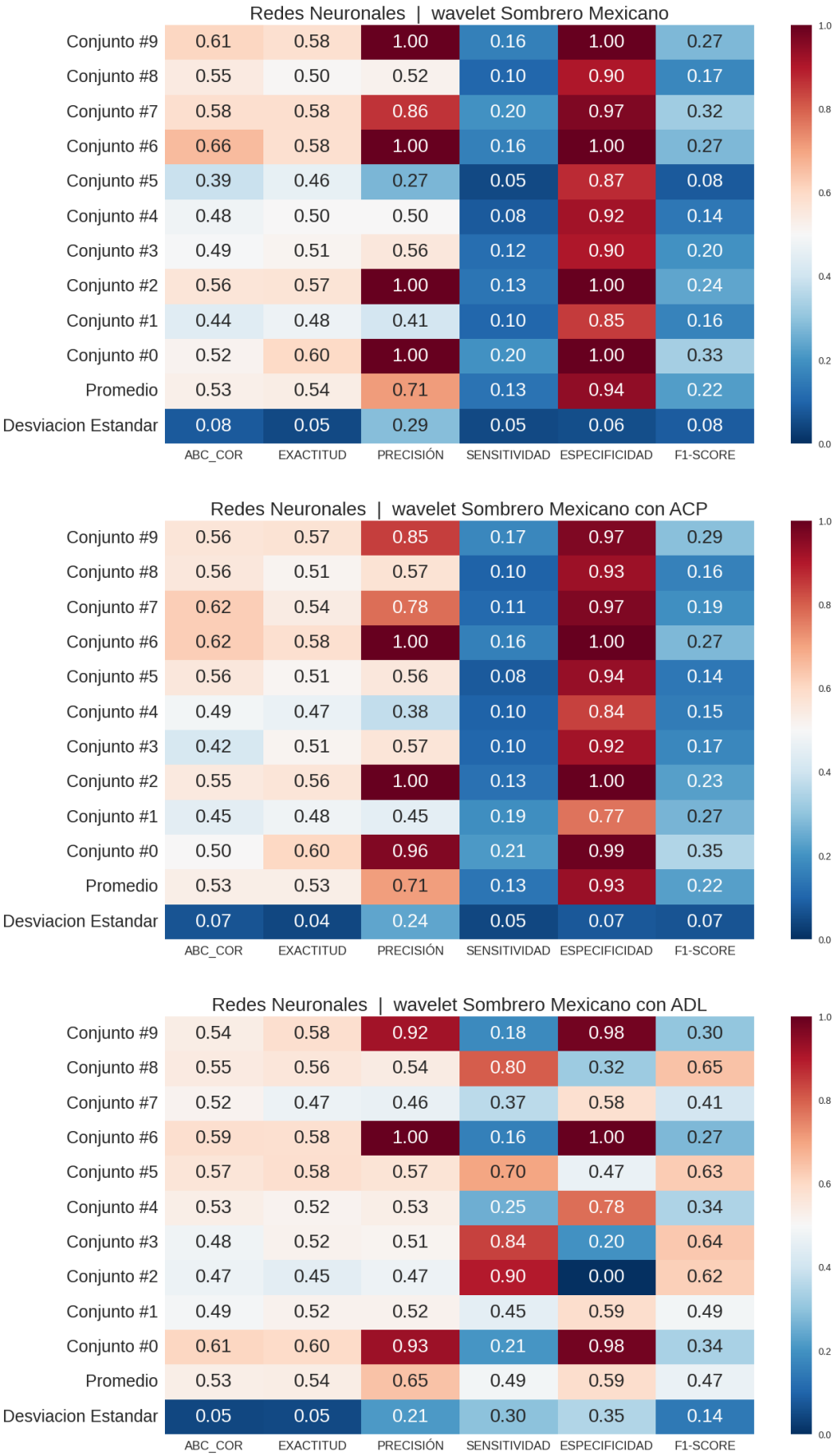


Figura A.4: Redes Neuronales con la wavelet Sombrero Mexicano.

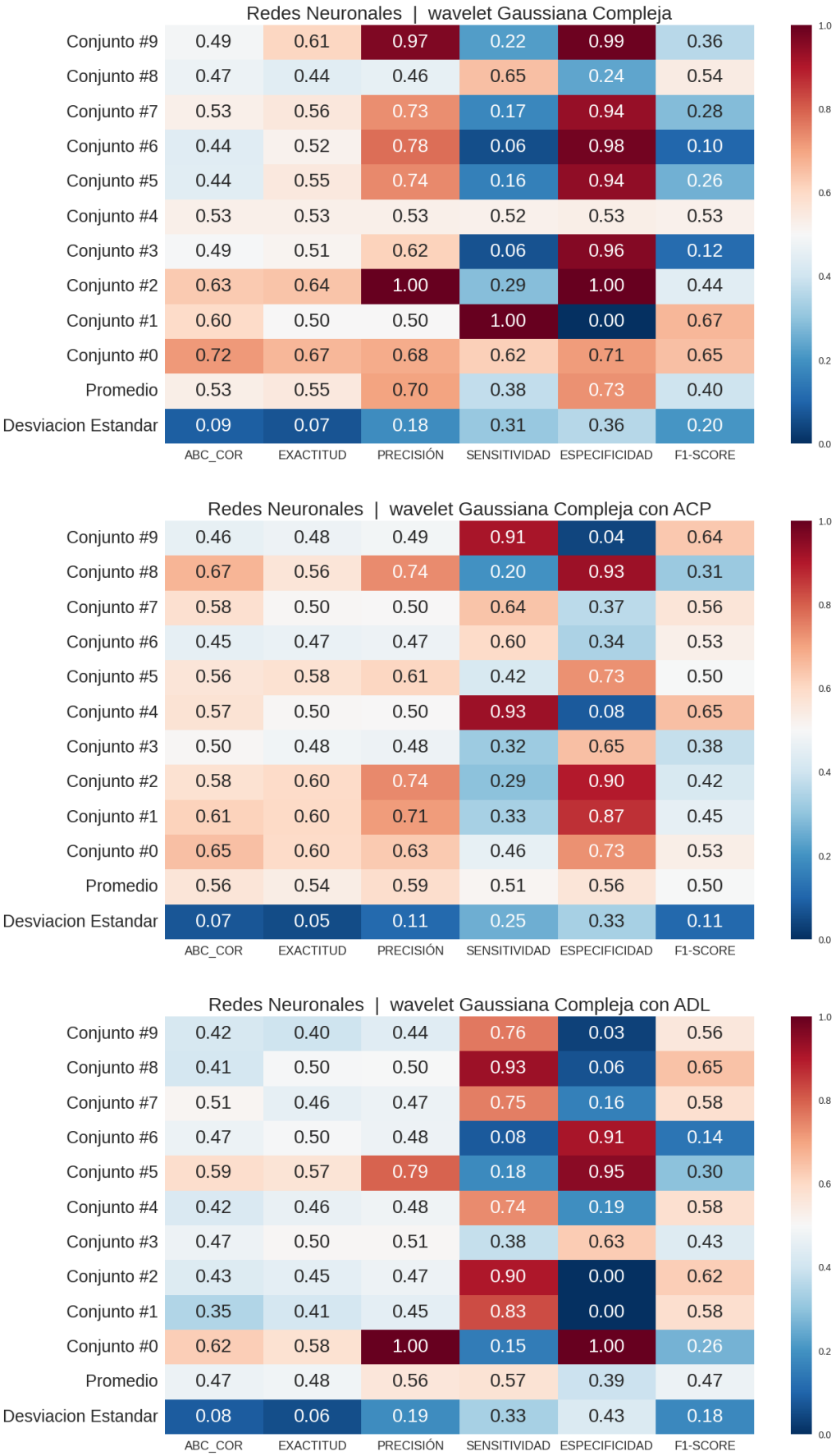


Figura A.5: Redes Neuronales con la wavelet Gaussiana Compleja.

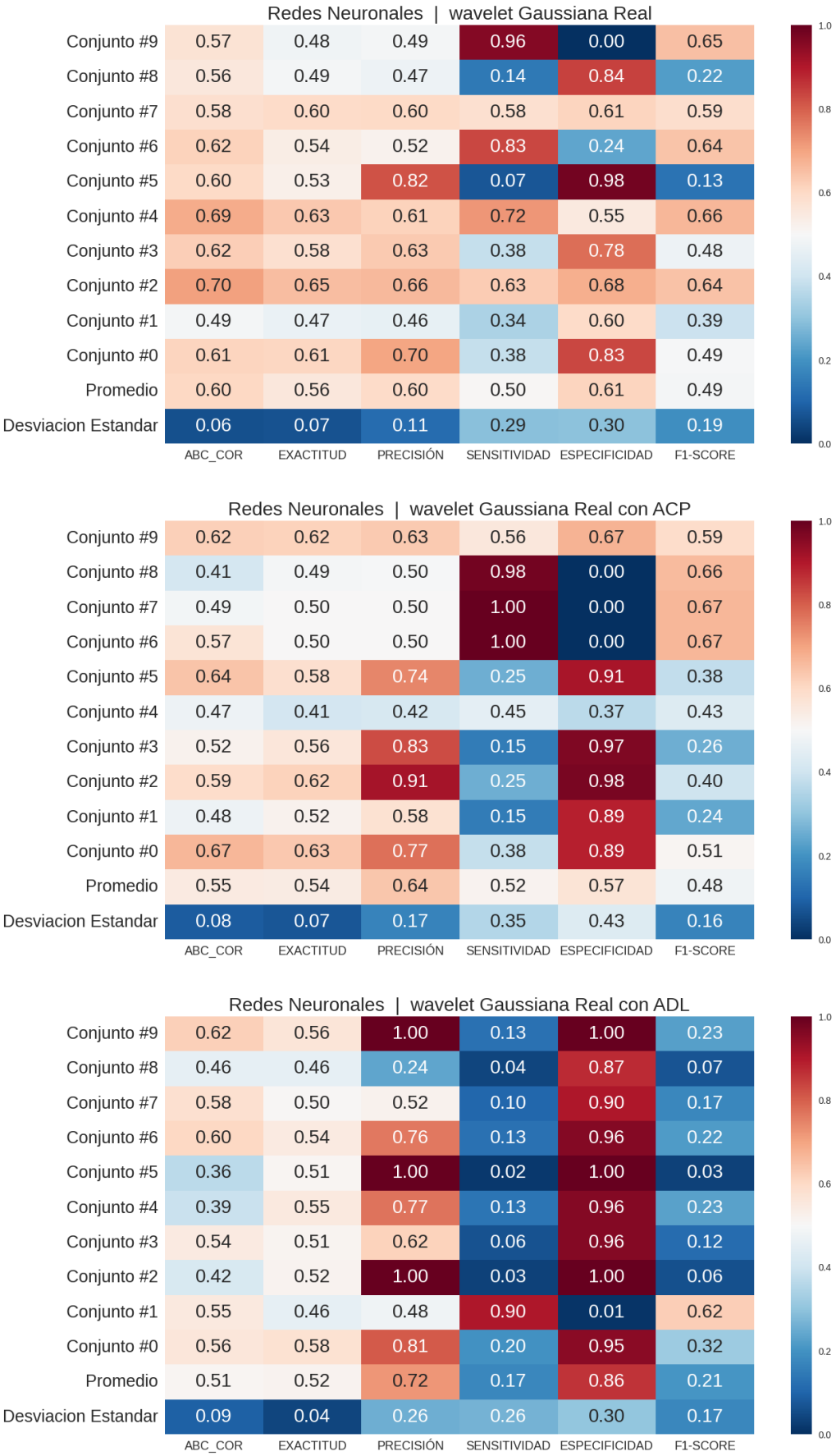


Figura A.6: Redes Neuronales con la wavelet Gaussiana Real.

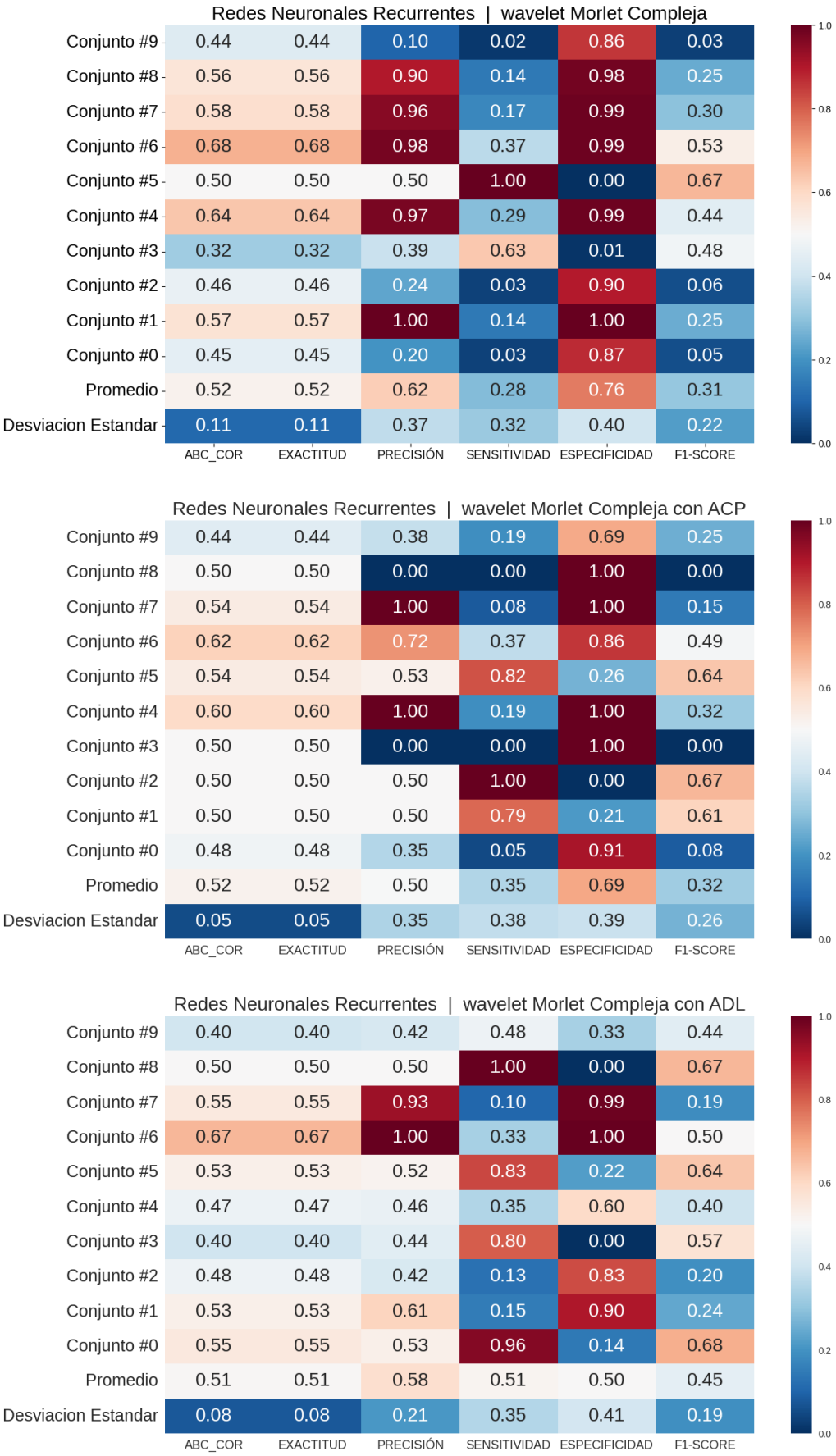


Figura A.7: Redes Neuronales Recurrentes con la wavelet Morlet Compleja.

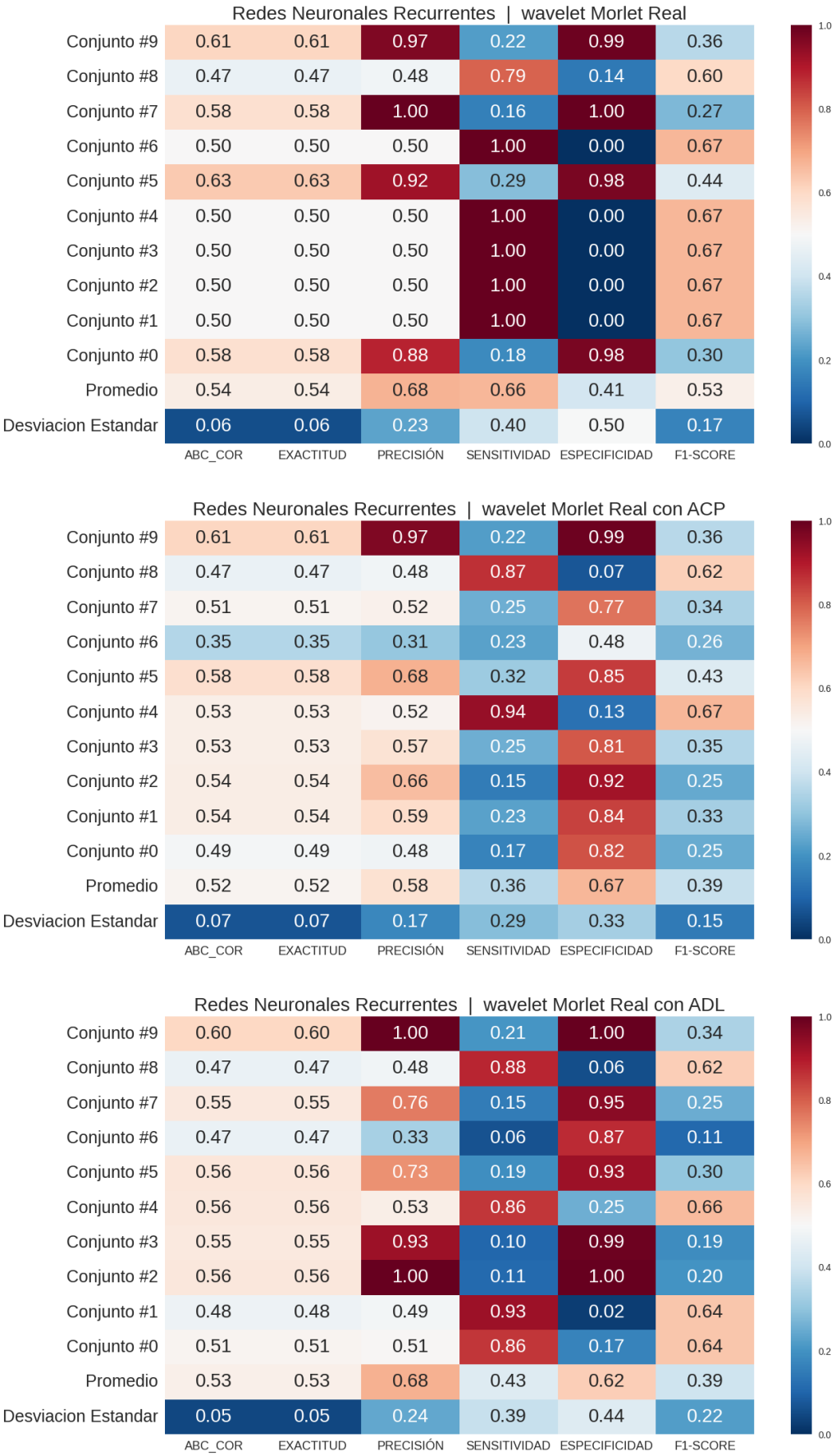


Figura A.8: Redes Neuronales Recurrentes con la wavelet Morlet Real.

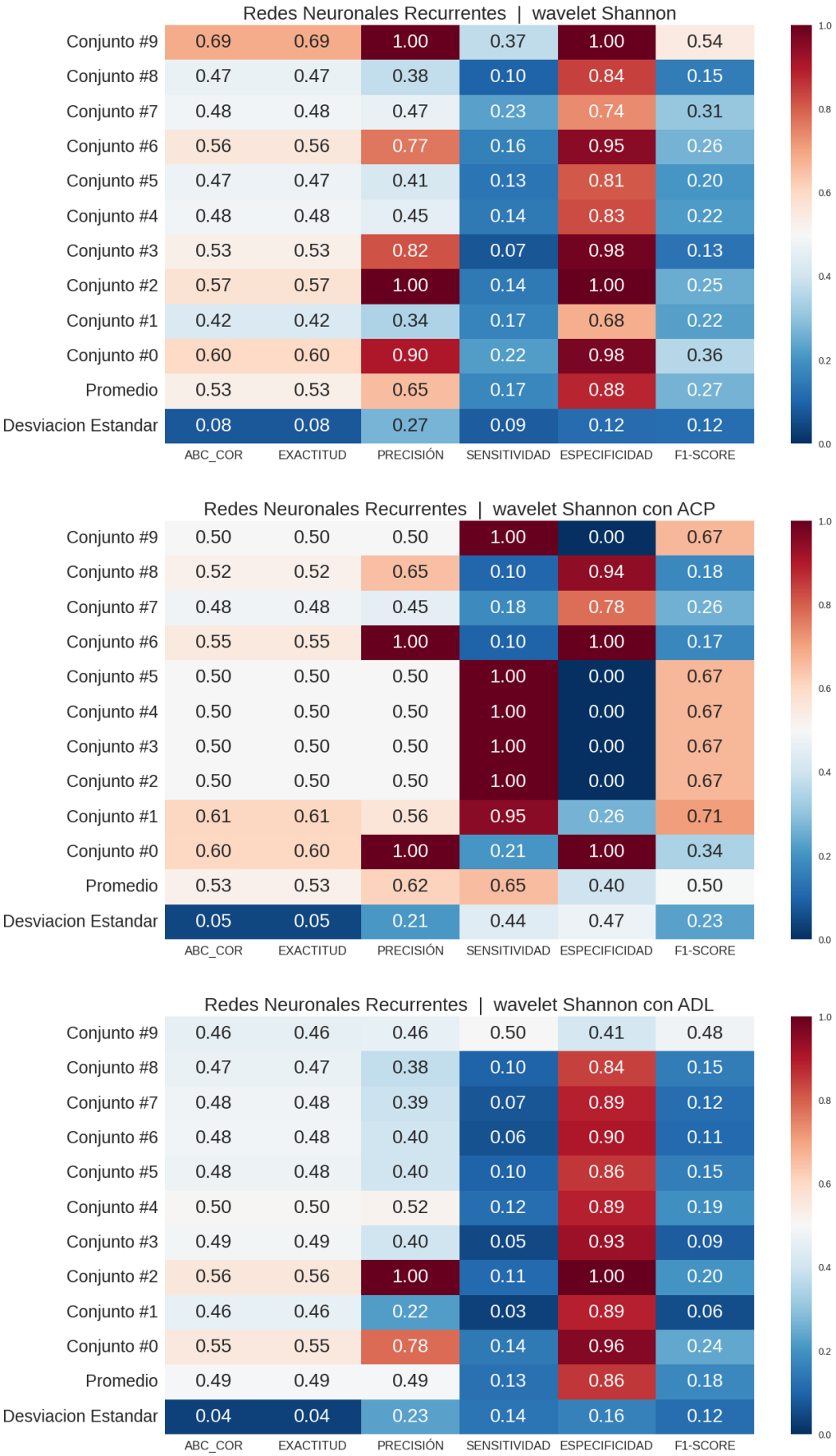


Figura A.9: Redes Neuronales Recurrentes con la wavelet Shannon.

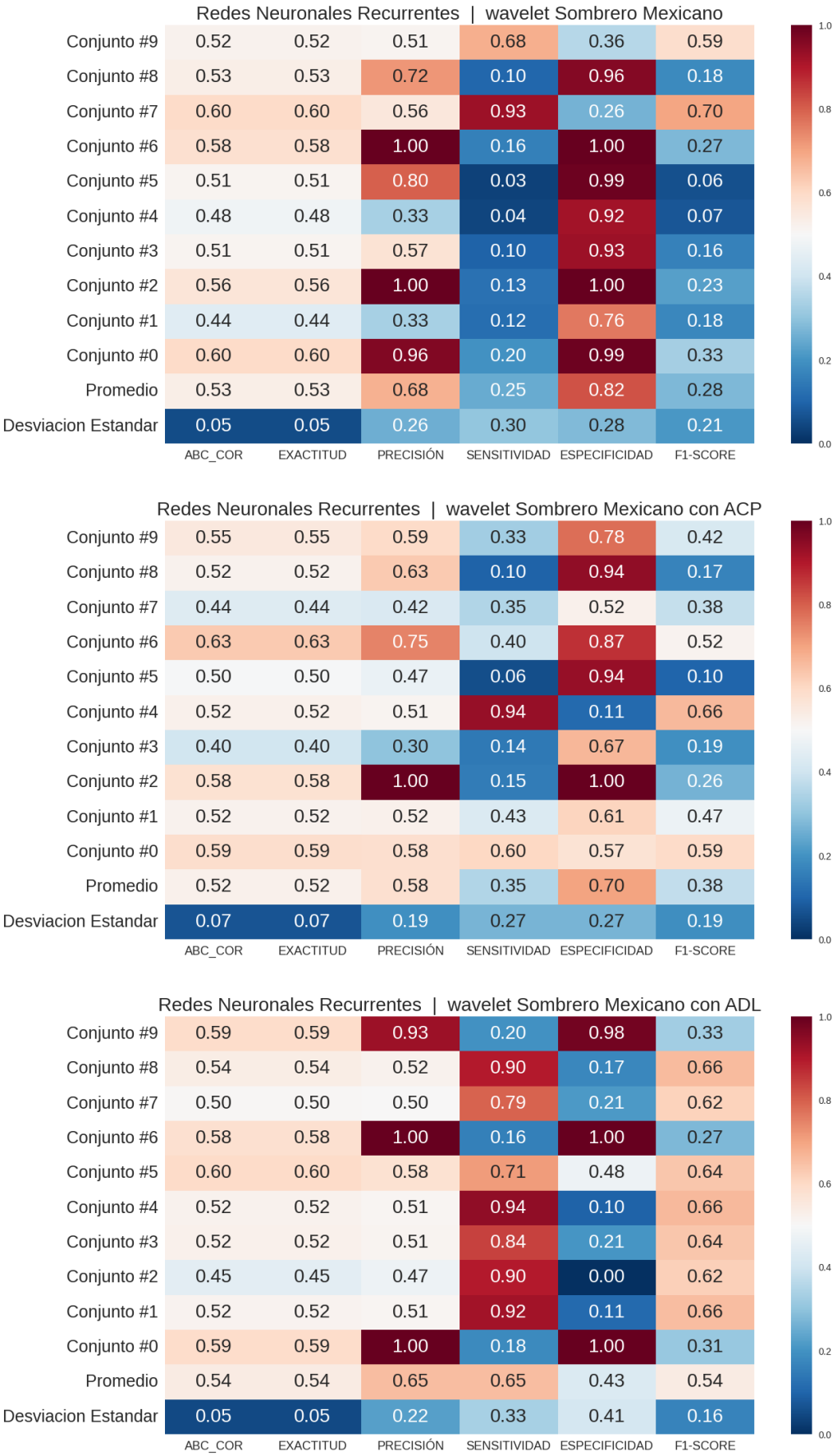


Figura A.10: Redes Neuronales Recurrentes con la wavelet Sombrero Mexicano.

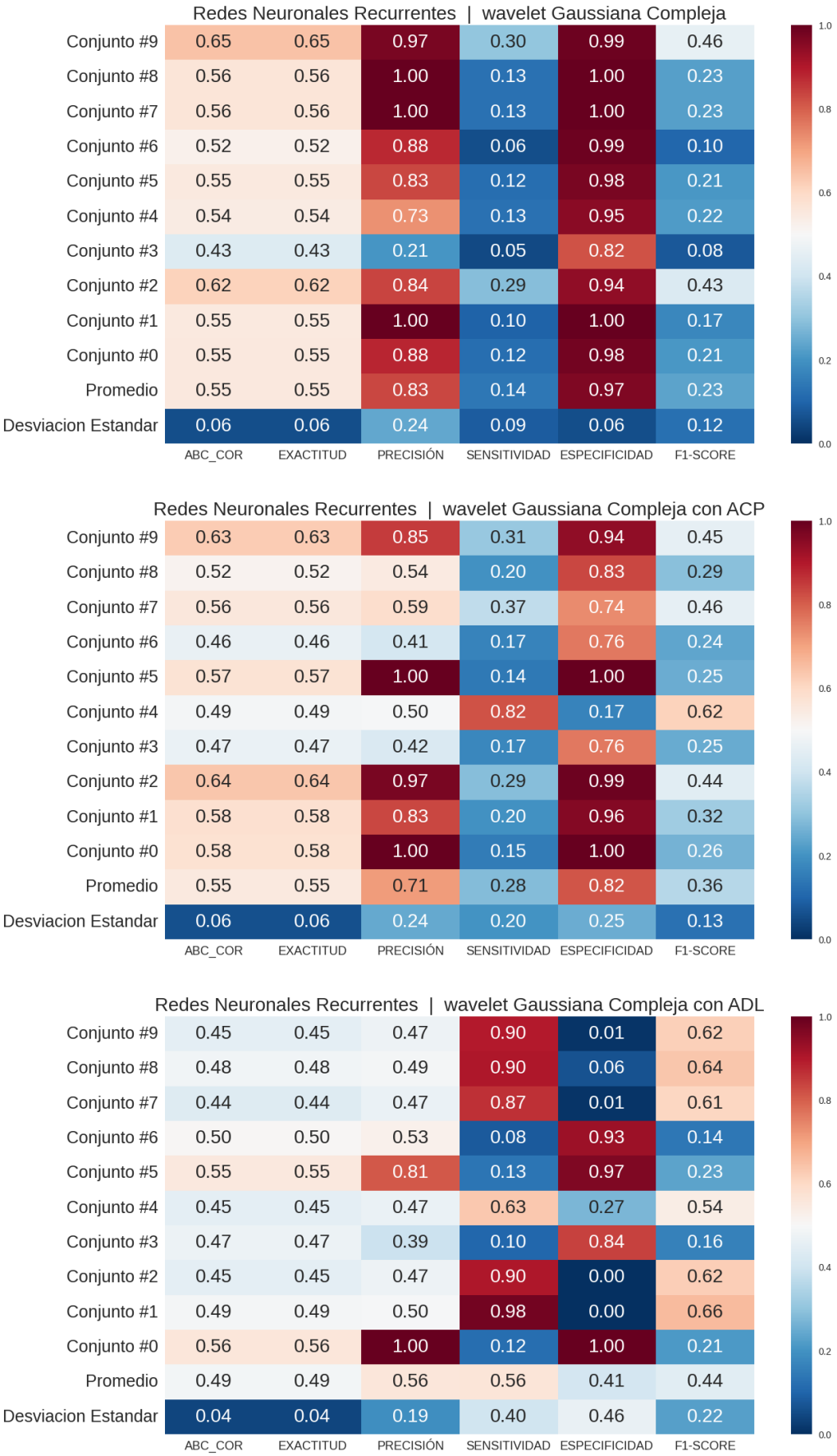


Figura A.11: Redes Neuronales Recurrentes con la wavelet Gaussiana Compleja.

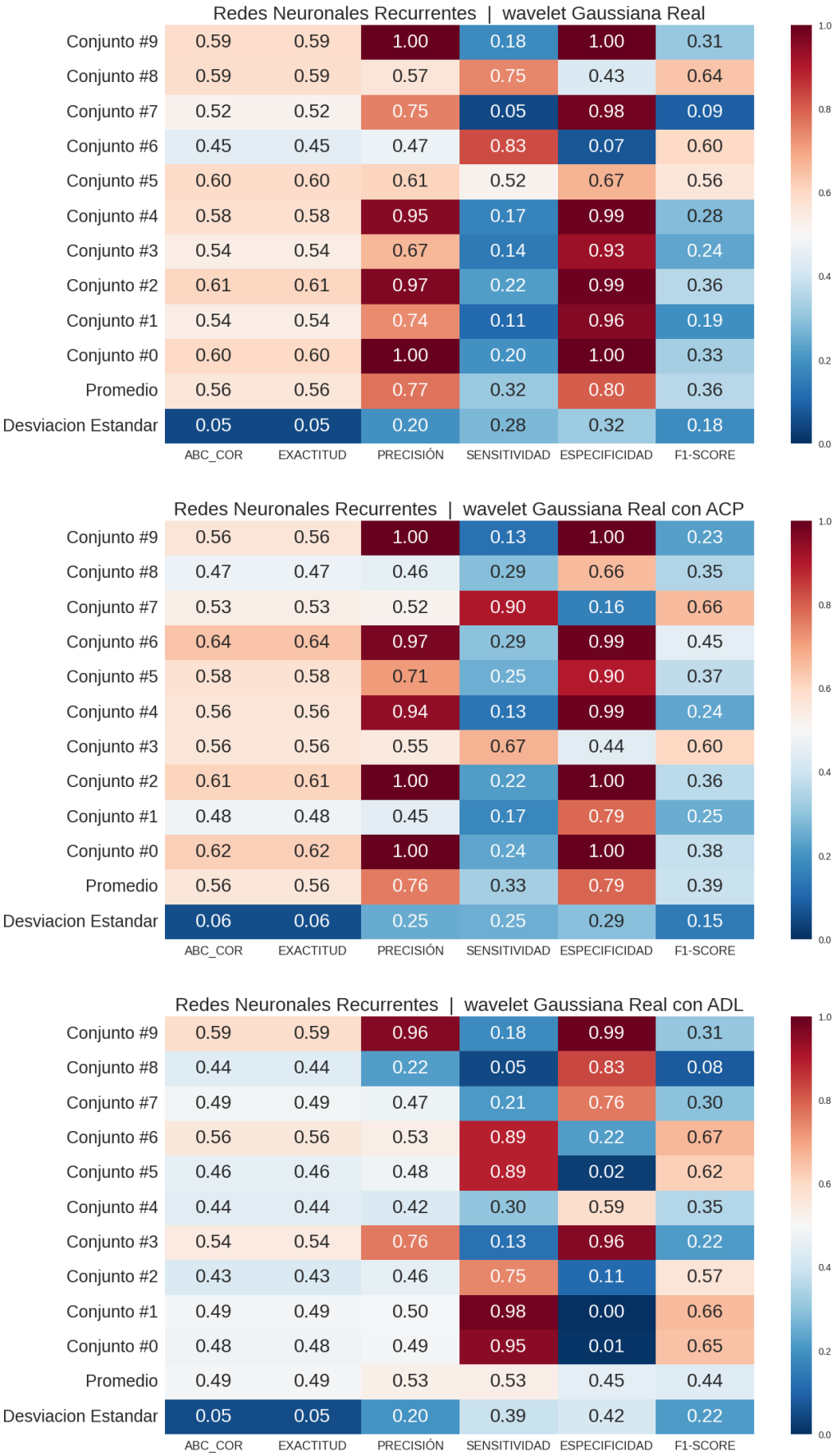


Figura A.12: Redes Neuronales Recurrentes con la wavelet Gaussiana Real.

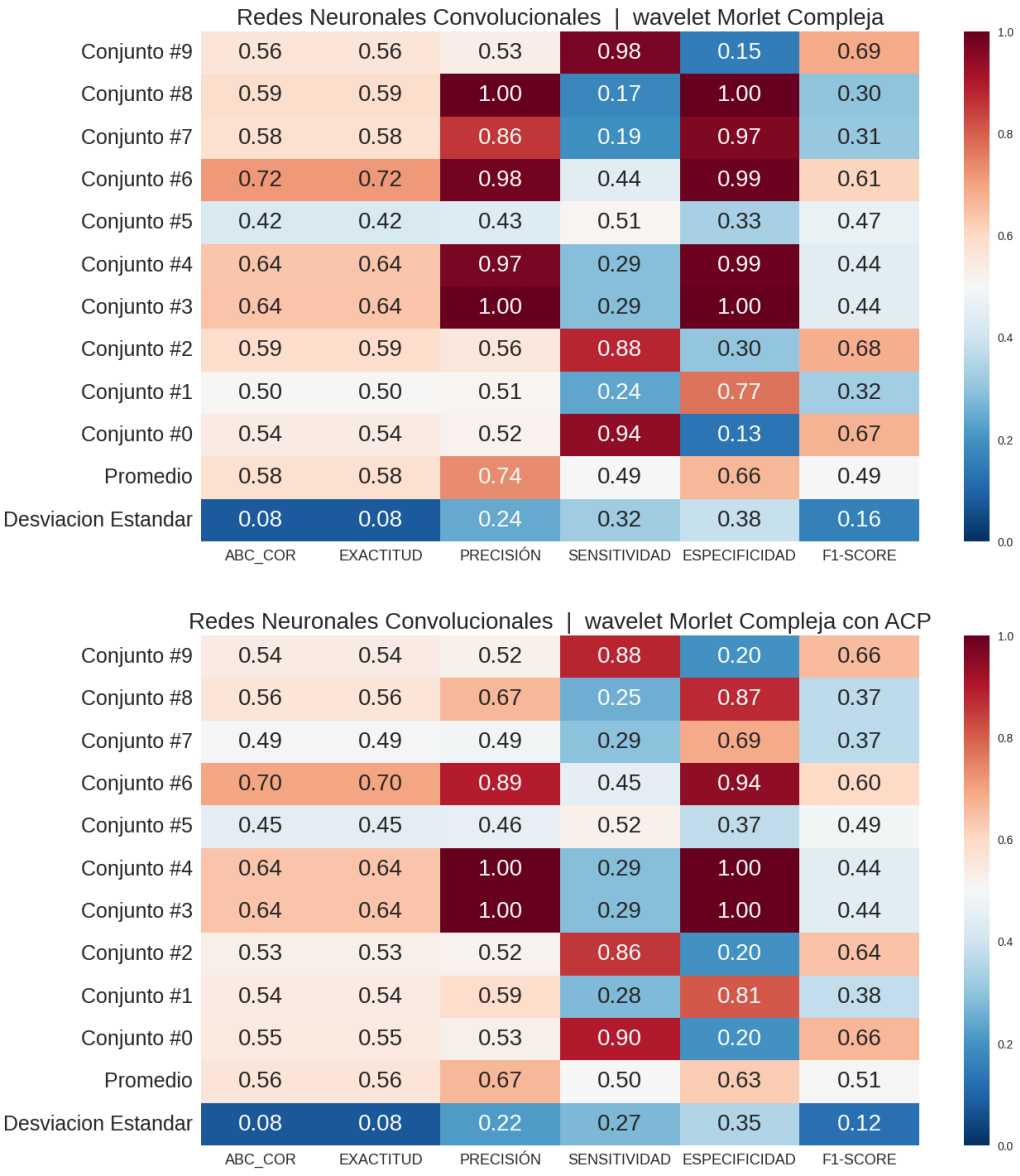


Figura A.13: Redes Neuronales Convolucionales con la wavelet Morlet Compleja.

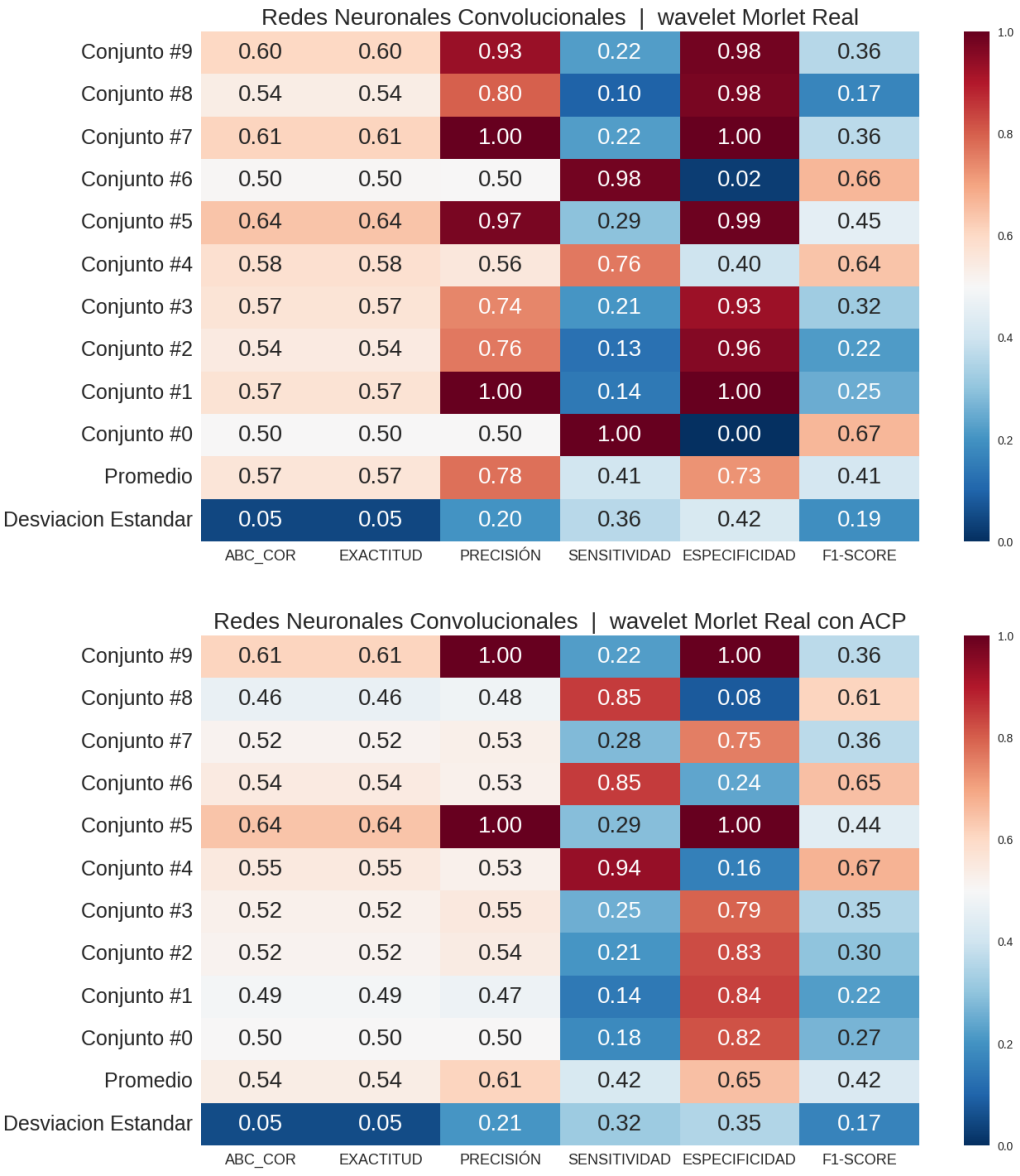


Figura A.14: Redes Neuronales Convolucionales con la wavelet Morlet Real.

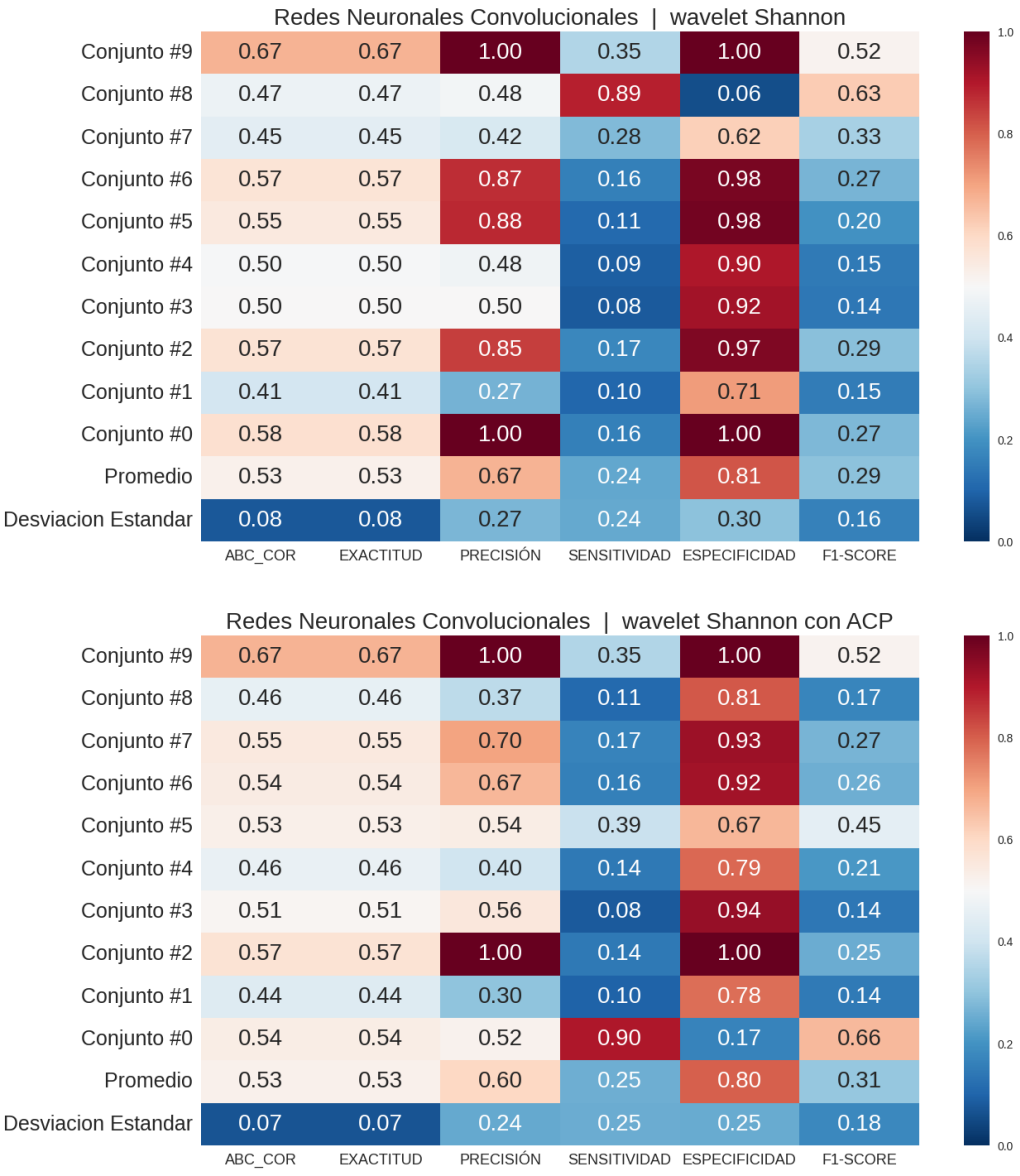


Figura A.15: Redes Neuronales Convolucionales con la wavelet Shannon.

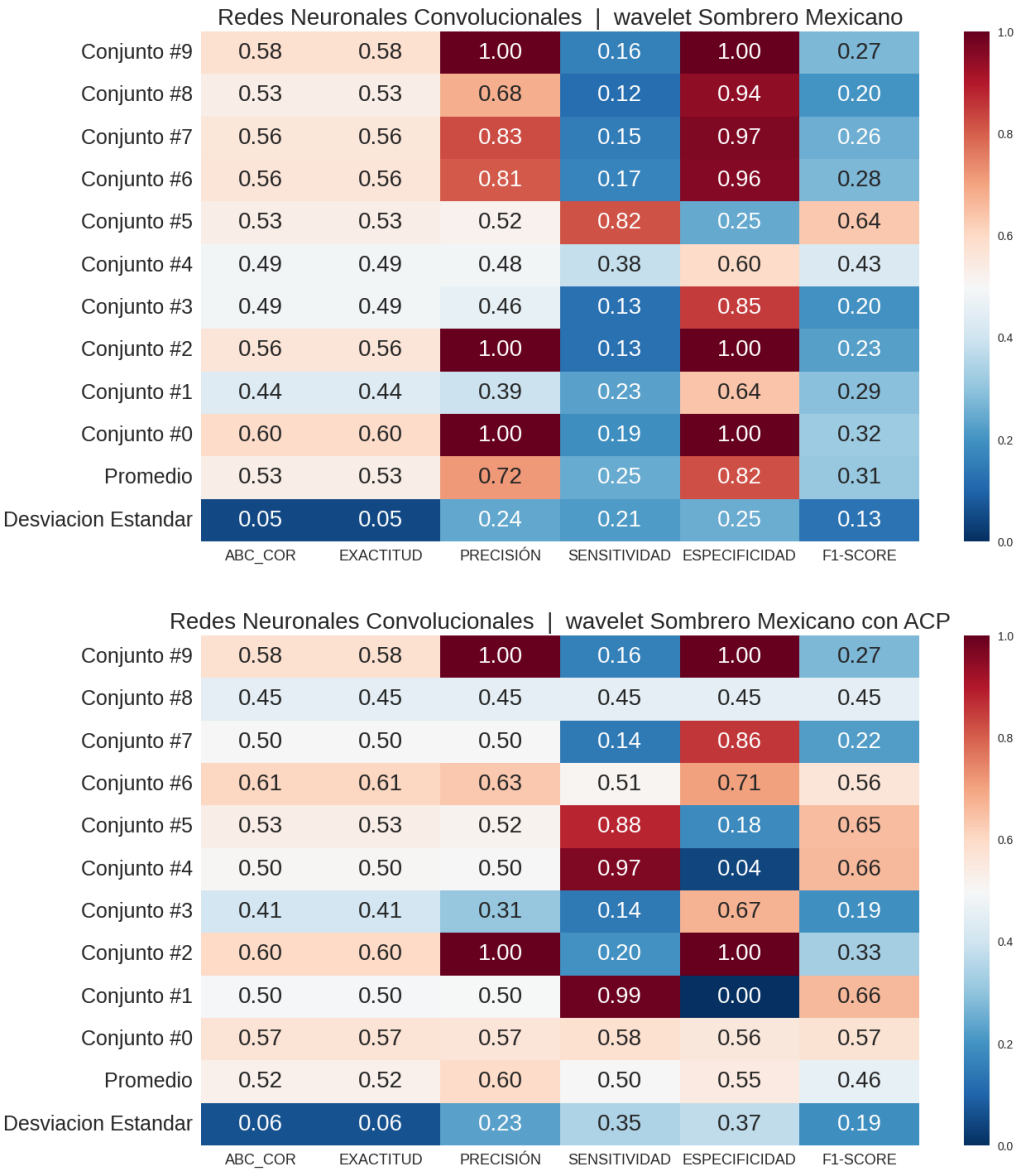


Figura A.16: Redes Neuronales Convolucionales con la wavelet Sombrero Mexicano.

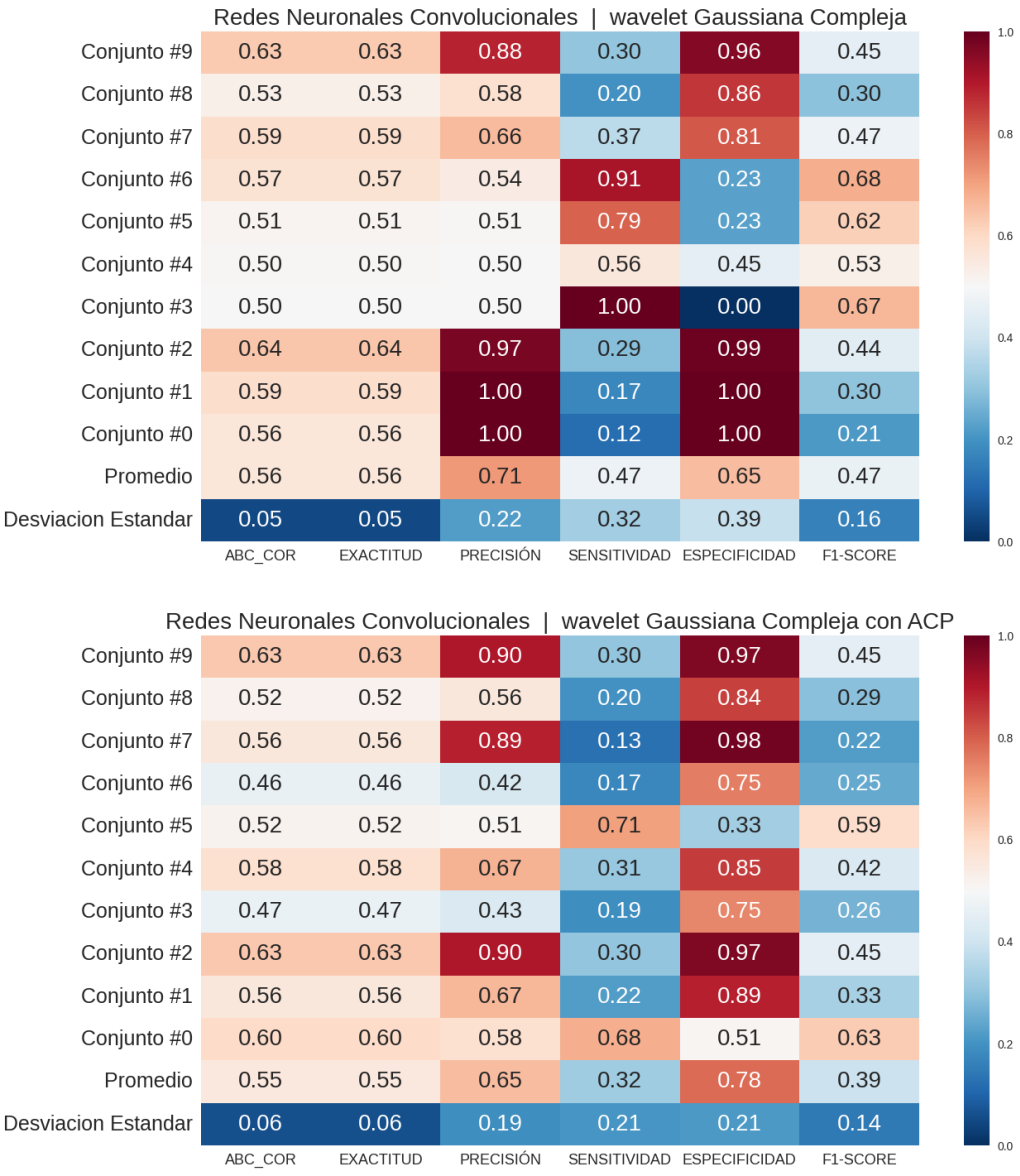


Figura A.17: Redes Neuronales Convolucionales con la wavelet Gaussiana Compleja.

Figura A.18: Redes Neuronales Convolucionales con la wavelet Gaussiana Real.

Apéndice B

Valores de hiperparámetros obtenidos

Cuadro B.1: **Hiperparámetros obtenidos para la wavelet Morlet Compleja**

		Valores de la optimización		
Algoritmo	**Hiperparámetros**	**Originales**	**ACP**	**ADL**
Bosques Aleatorios	Estado aleatorio	124	981	577
	No. estimadores	9	7	83
	Profundidad máxima del árbol	default	default	default
	Número de muestras para entrenar cada estimador	577	647	704
	Número mínimo de muestras para dividir un nodo	64	153	36
	Muestras mínimas para ser un nodo hoja	143	141	17
Bayes Ingenuo Multinomial	Calcular probabilidades a priori	falso	-	-
	α del suavizado de Laplace	0.50136	-	-
Bayes Ingenuo Gaussiano	Estabilizador de la varianza	0.70457	0.84288	0.86835
Redes Neuronales	Estado alestorio	232	131	87
	Número de capas	2	1	2
	Perceptrones por capa	75, 44	10	15, 15
	Función de activación	logística	logística	tanh
	Algoritmo de optimización	adam	sgd	lbfgs
	α para Regularización L2	0.00414	0.00217	0.00043
	Máximo de iteraciones	1000	1000	1000
Redes Neuronales Convolucionales	Número de filtros y tamaño de la ventana de convolución	1 y 1	4 y 2	-
	Desplazamiento de la ventana de convolución	4	1	-
	Función de activación	elu	tanh	-
	Algoritmo de optimización	adam	adam	-
	Tasa de aprendizaje	0.00229	0.00196	-
	Épocas	1000	1000	-
Redes Neuronales Recurrentes	Unidades	1	2	4 y 2
	Función de activación	elu	softplus	relu
	Algoritmo de optimización	adam	adam	adam
	Tasa de aprendizaje	0.00320	0.4353	0.00004
	Épocas	1000	1000	1000

Cuadro B.2: **Hiperparámetros obtenidos para la wavelet Morlet Real**

		Valores de la optimización		
Algoritmo	**Hiperparámetros**	**Originales**	**ACP**	**ADL**
Bosques Aleatorios	Estado aleatorio	219	460	771
	No. estimadores	38	156	2
	Profundidad máxima del árbol	default	default	default
	Número de muestras para entrenar cada estimador	191	779	284
	Número mínimo de muestras para dividir un nodo	21	12	80
	Muestras mínimas para ser un nodo hoja	87	6	52
Bayes Ingenuo Multinomial	Calcular probabilidades a priori	verdadero	-	-
	α del suavizado de Laplace	0.33060	-	-
Bayes Ingenuo Gaussiano	Estabilizador de la varianza	0.42201	0.10402	0.14774
Redes Neuronales	Estado aleatorio	248	125	151
	Número de capas	3	3	4
	Perceptrones por capa	99, 99, 40	75, 47, 47	13, 88, 56, 19
	Función de activación	identidad	relu	logística
	Algoritmo de optimización	lbfgs	adam	sgd
	α para Regularización L2	0.00217	0.00217	0.00323
	Máximo de iteraciones	1000	1000	1000
Redes Neuronales Convolucionales	Número de filtros y tamaño de la ventana de convolución	3 y 2	2 y 2	-
	Desplazamiento de la ventana de convolución	1	4	-
	Función de activación	exponencial	tanh	-
	Algoritmo de optimización	adam	adam	-
	Tasa de aprendizaje	0.00983	0.00927	-
	Épocas	1000	1000	-
Redes Neuronales Recurrentes	Unidades	1	4	1
	Función de activación	exponencial	softplus	exponencial
	Algoritmo de optimización	adam	adam	adam
	Tasa de aprendizaje	0.00800	0.00126	0.00164
	Épocas	1000	1000	1000

Cuadro B.3: **Hiperparámetros obtenidos para la wavelet Gaussiana Compleja**

		Valores de la optimización		
Algoritmo	**Hiperparámetros**	**Originales**	**ACP**	**ADL**
Bosques Aleatorios	Estado aleatorio	624	691	115
	No. estimadores	7	1	17
	Profundidad máxima del árbol	default	default	default
	Número de muestras para entrenar cada estimador	654	391	780
	Número mínimo de muestras para dividir un nodo	68	100	2
	Muestras mínimas para ser un nodo hoja	41	43	6
Bayes Ingenuo Multinomial	Calcular probabilidades a priori	falso	-	-
	α del suavizado de Laplace	0.99843	-	-
Bayes Ingenuo Gaussiano	Estabilizador de la varianza	0.00473	0.00814	0.01374
Redes Neuronales	Estado aleatorio	69	90	264
	Número de capas	3	3	3
	Perceptrones por capa	1, 36, 60	23, 42, 53	18, 31, 35
	Función de activación	relu	tanh	tanh
	Algoritmo de optimización	lbfgs	lbfgs	lbfgs
	α para Regularización L2	0.00433	0.00186	0.00062
	Máximo de iteraciones	1000	1000	1000
Redes Neuronales Convolucionales	Número de filtros y tamaño de la ventana de convolución	1 y 2	2 y 1	-
	Desplazamiento de la ventana de convolución	3	1	-
	Función de activación	exponencial	sigmoide	-
	Algoritmo de optimización	adam	adam	-
	Tasa de aprendizaje	0.00887	0.00361	-
	Épocas	1000	1000	-
Redes Neuronales Recurrentes	Unidades	4	4	3
	Función de activación	elu	relu	exponencial
	Algoritmo de optimización	adam	adam	adam
	Tasa de aprendizaje	0.00285	0.00110	0.00922
	Épocas	1000	1000	1000

Cuadro B.4: **Hiperparámetros obtenidos para la wavelet Gaussiana Real**

		Valores de la optimización		
Algoritmo	**Hiperparámetros**	**Originales**	**ACP**	**ADL**
Bosques Aleatorios	Estado aleatorio	432	53	22
	No. de estimadores o árboles de decisión utilizados	4	10	3
	Profundidad máxima del árbol	default	default	default
	Número de muestras para entrenar cada estimador	331	824	162
	Número mínimo de muestras para dividir un nodo	248	40	65
	Muestras mínimas para ser un nodo hoja	72	14	12
Bayes Ingenuo Multinomial	Calcular probabilidades a priori	falso	-	-
	α del suavizado de Laplace	0.72599	-	-
Bayes Ingenuo Gaussiano	Estabilizador de la varianza	0.01368	0.01405	0.96828
Redes Neuronales	Estado aleatorio	289	42	279
	Número de capas	4	2	2
	Perceptrones por capa	100, 48, 40, 70	87, 28	22, 18
	Función de activación	relu	logística	logística
	Algoritmo de optimización	lbfgs	sgd	sgd
	α para Regularización L2	0.00210	0.00292	0.00465
	Máximo de iteraciones	1000	1000	1000
Redes Neuronales Convolucionales	Número de filtros y tamaño de la ventana de convolución	3 y 3	1 y 1	-
	Desplazamiento de la ventana de convolución	1	2	-
	Función de activación	softplus	exponencial	-
	Algoritmo de optimización	adam	adam	-
	Tasa de aprendizaje	0.00865	0.00363	-
	Épocas	1000	1000	-
Redes Neuronales Recurrentes	Unidades	2	2	2
	Función de activación	tanh	selu	exponencial
	Algoritmo de optimización	adam	adam	adam
	Tasa de aprendizaje	0.00991	0.00099	0.00688
	Épocas	1000	1000	1000

Cuadro B.5: **Hiperparámetros obtenidos para la wavelet Shannon**

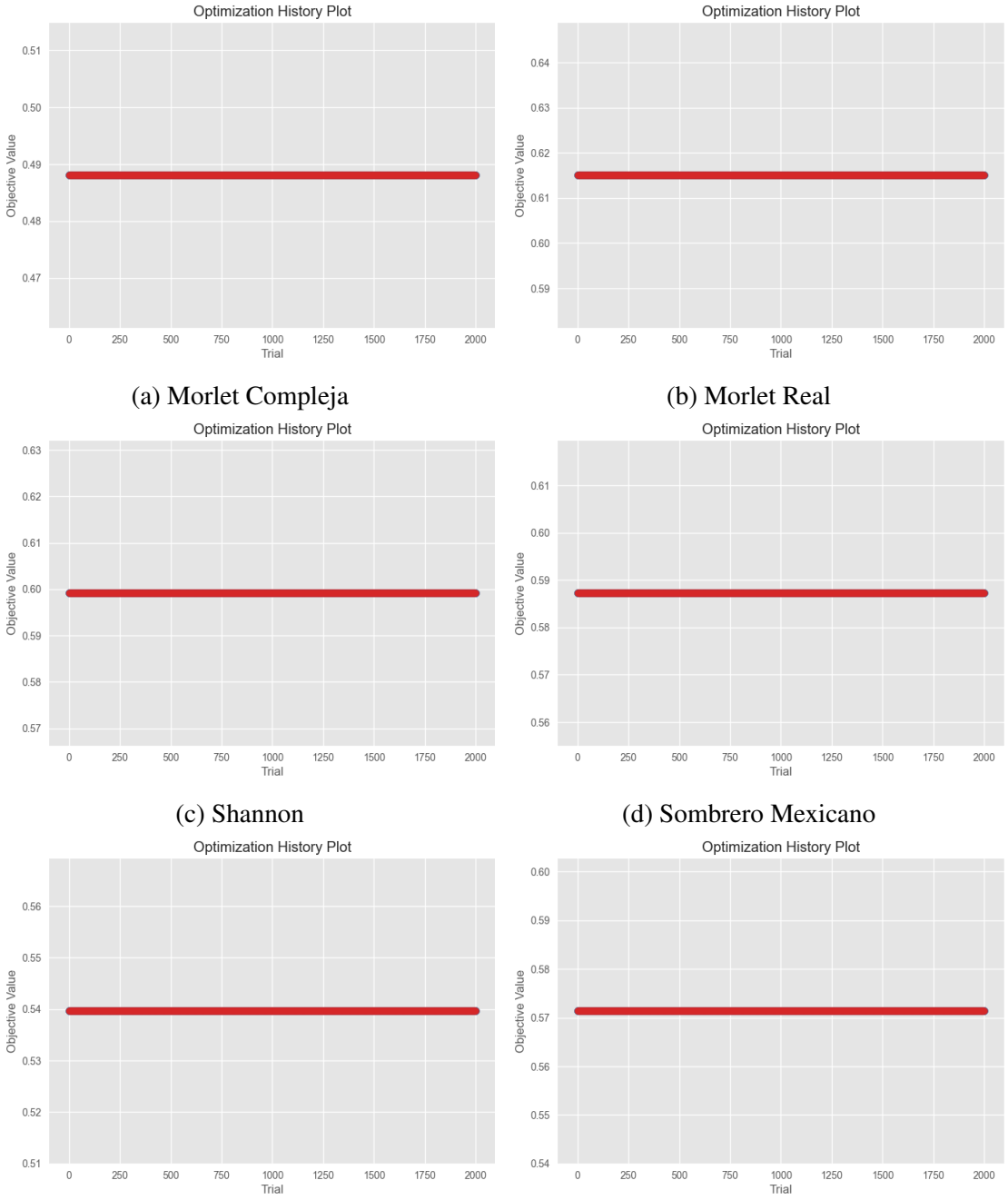
		Valores de la optimización		
Algoritmo	**Hiperparámetros**	**Originales**	**ACP**	**ADL**
Bosques Aleatorios	Estado aleatorio	704	152	49
	No. estimadores	26	38	243
	Profundidad máxima del árbol	default	default	default
	Número de muestras para entrenar cada estimador	524	291	920
	Número mínimo de muestras para dividir un nodo	48	240	265
	Muestras mínimas para ser un nodo hoja	37	54	167
Bayes Ingenuo Multinomial	Calcular probabilidades a priori	falso	-	-
	α del suavizado de Laplace	0.91247	-	-
Bayes Ingenuo Gaussiano	Estabilizador de la varianza	0.70020	0.12562	0.17495
Redes Neuronales	Estado aleatorio	118	33	283
	Número de capas	2	1	2
	Perceptrones por capa	39, 8	43	78, 100
	Función de activación	relu	logística	relu
	Algoritmo de optimización	lbfgs	sgd	lbfgs
	α para Regularización L2	0.00451	0.00053	0.00451
	Máximo de iteraciones	1000	1000	1000
Redes Neuronales Convolucionales	Número de filtros y tamaño de la ventana de convolución	1 y 4	2 y 1	-
	Desplazamiento de la ventana de convolución	2	1	-
	Función de activación	exponencial	tanh	-
	Algoritmo de optimización	adam	adam	-
	Tasa de aprendizaje	0.00665	0.00179	-
	Épocas	1000	1000	-
Redes Neuronales Recurrentes	Unidades	2	1	3
	Función de activación	selu	exponencial	elu
	Algoritmo de optimización	adam	adam	adam
	Tasa de aprendizaje	0.00114	0.00427	0.00783
	Épocas	1000	1000	1000

Cuadro B.6: **Hiperparámetros obtenidos para la wavelet Sombrero Mexicano**

		Valores de la optimización		
Algoritmo	**Hiperparámetros**	**Originales**	**ACP**	**ADL**
Bosques Aleatorios	Estado aleatorio	928	281	443
	No. de estimadores o árboles de decisión utilizados	11	7	12
	Profundidad máxima del árbol	default	default	default
	Número de muestras para entrenar cada estimador	710	674	216
	Número mínimo de muestras para dividir un nodo	32	44	196
	Muestras mínimas para ser un nodo hoja	158	15	16
Bayes Ingenuo Multinomial	Calcular probabilidades a priori	falso	-	-
	α del suavizado de Laplace	0.12255	-	-
Bayes Ingenuo Gaussiano	Estabilizador de la varianza	0.01135	0.12512	0.94404
Redes Neuronales	Estado aleatorio	177	257	127
	Número de capas	2	2	4
	Perceptrones por capa	23, 1	22, 33	37, 46, 59, 34
	Función de activación	logística	logística	logística
	Algoritmo de optimización	sgd	sgd	lbfgs
	α para Regularización L2	0.00396	0.00472	0.00248
	Máximo de iteraciones	1000	1000	1000
Redes Neuronales Convolucionales	Número de filtros y tamaño de la ventana de convolución	4 y 1	1 y 1	-
	Desplazamiento de la ventana de convolución	3	1	-
	Función de activación	sigmoide	elu	-
	Algoritmo de optimización	adam	adam	-
	Tasa de aprendizaje	0.00026	0.00640	-
	Épocas	1000	1000	-
Redes Neuronales Recurrentes	Unidades	4	3	2
	Función de activación	relu	tanh	softsign
	Algoritmo de optimización	adam	adam	adam
	Tasa de aprendizaje	0.00493	0.00175	0.00358
	Épocas	1000	1000	1000

Apéndice C

Gráficas de optimización de hiperparámetros



(a) Morlet Compleja

(b) Morlet Real

(c) Shannon

(d) Sombrero Mexicano

(e) Gaussiana Compleja

(f) Gaussiana Real

Figura C.1: Gráficas de optimización de hiperparámetros para Bayes Ingenuo Multinomial con todas las wavelets.

(a) Morlet Compleja

(b) Morlet Compleja con ACP

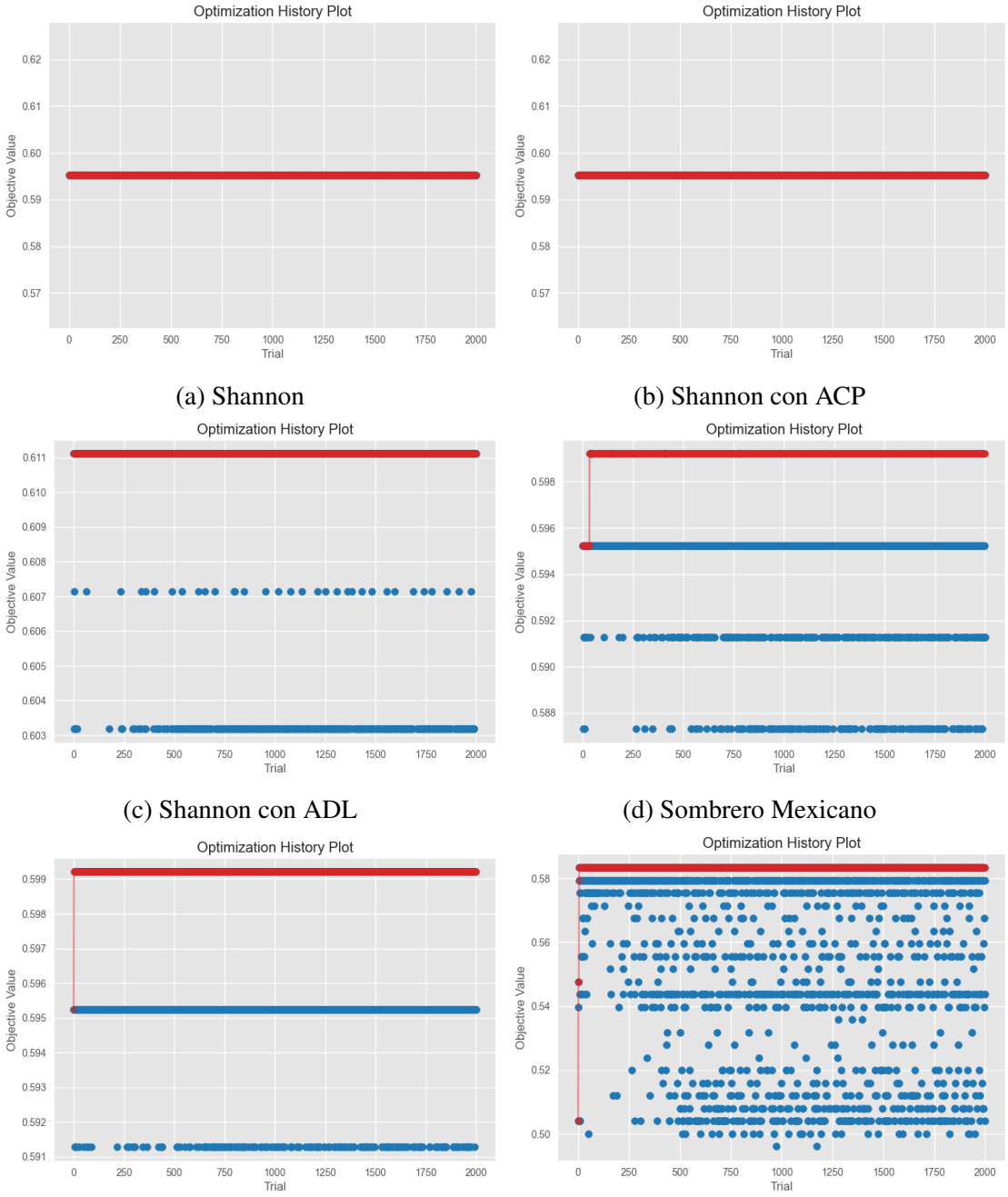
(c) Morlet Compleja con ADL

(d) Morlet Real

(e) Morlet Real con ACP

(f) Morlet Real con ADL

Figura C.2: Gráficas de optimización de hiperparámetros para Bayes Ingenuo Gaussiano con las wavelets Morlet Compleja y Morlet Real.



(a) Shannon

(b) Shannon con ACP

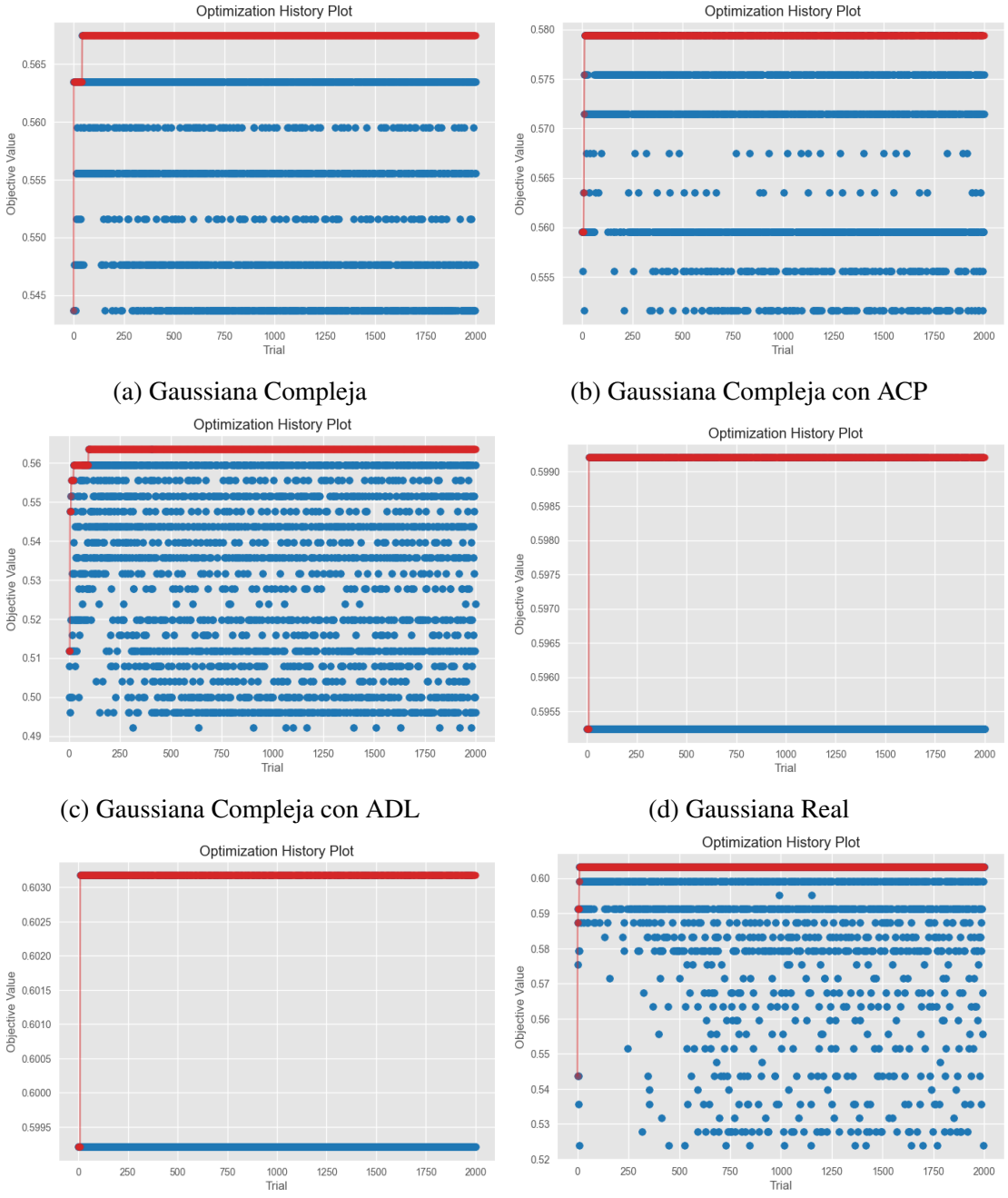
(c) Shannon con ADL

(d) Sombrero Mexicano

(e) Sombrero Mexicano con ACP

(f) Sombrero Mexicano con ADL

Figura C.3: Gráficas de optimización de hiperparámetros para Bayes Ingenuo Gaussiano con las wavelets Shannon y Sombrero Mexicano.



(a) Gaussiana Compleja

(b) Gaussiana Compleja con ACP

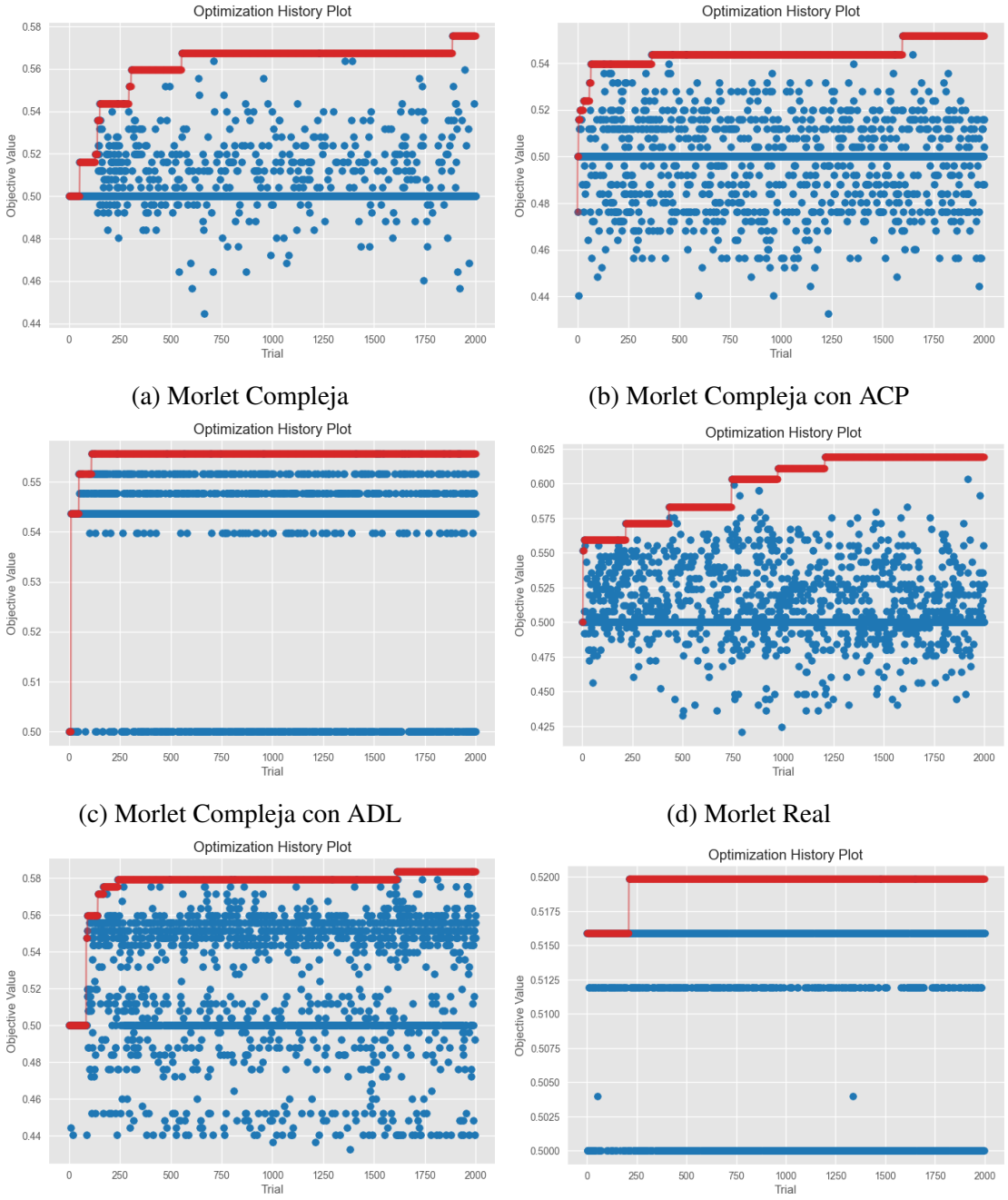
(c) Gaussiana Compleja con ADL

(d) Gaussiana Real

(e) Gaussiana Real con ACP

(f) Gaussiana Real con ADL

Figura C.4: Gráficas de optimización de hiperparámetros para Bayes Ingenuo Gaussiano con las wavelets Gaussiana Compleja y Gaussiana Real.



(a) Morlet Compleja

(b) Morlet Compleja con ACP

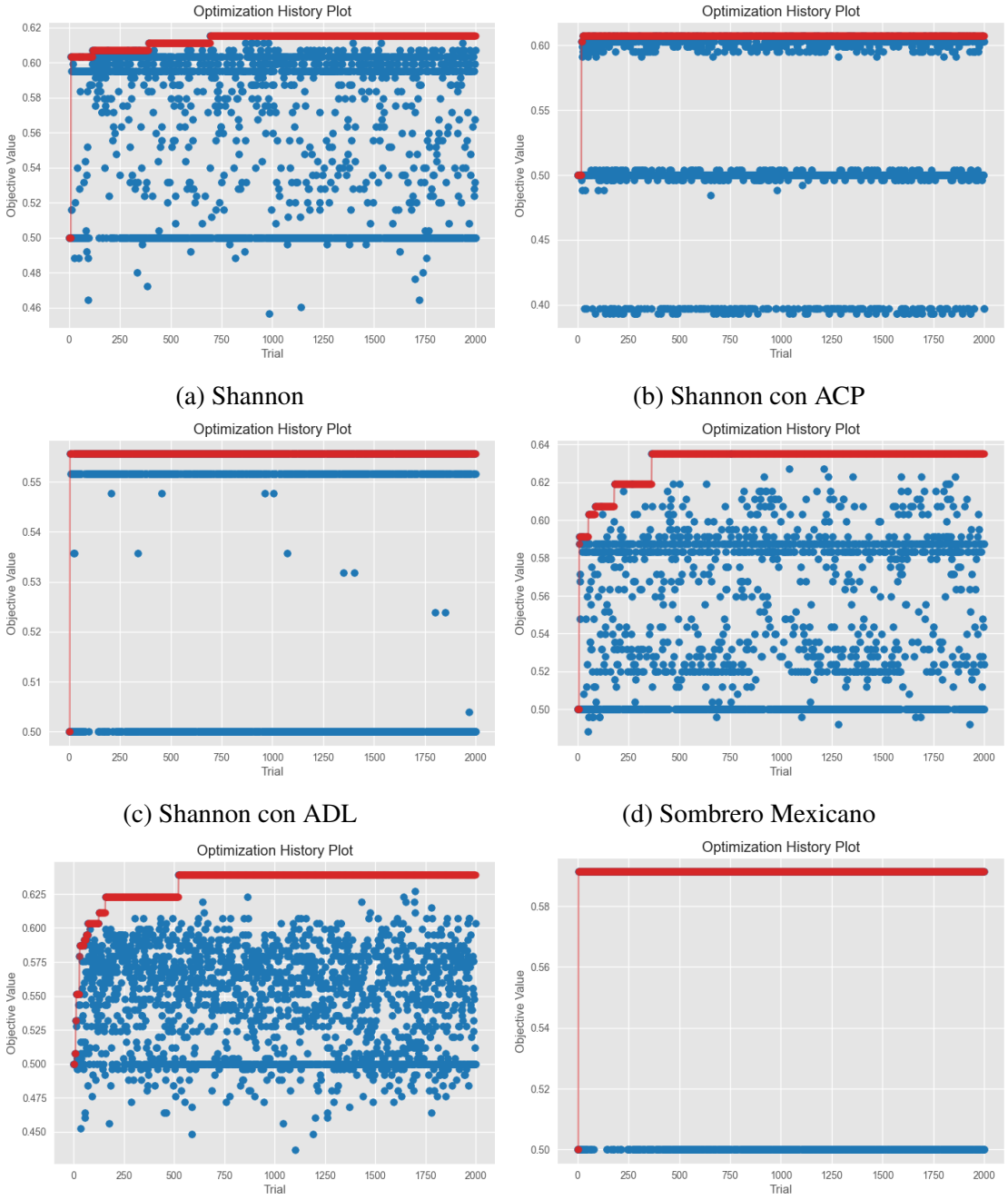
(c) Morlet Compleja con ADL

(d) Morlet Real

(e) Morlet Real con ACP

(f) Morlet Real con ADL

Figura C.5: Gráficas de optimización de hiperparámetros para Bosques Aleatorios con las wavelets Morlet Compleja y Morlet Real.



(a) Shannon

(b) Shannon con ACP

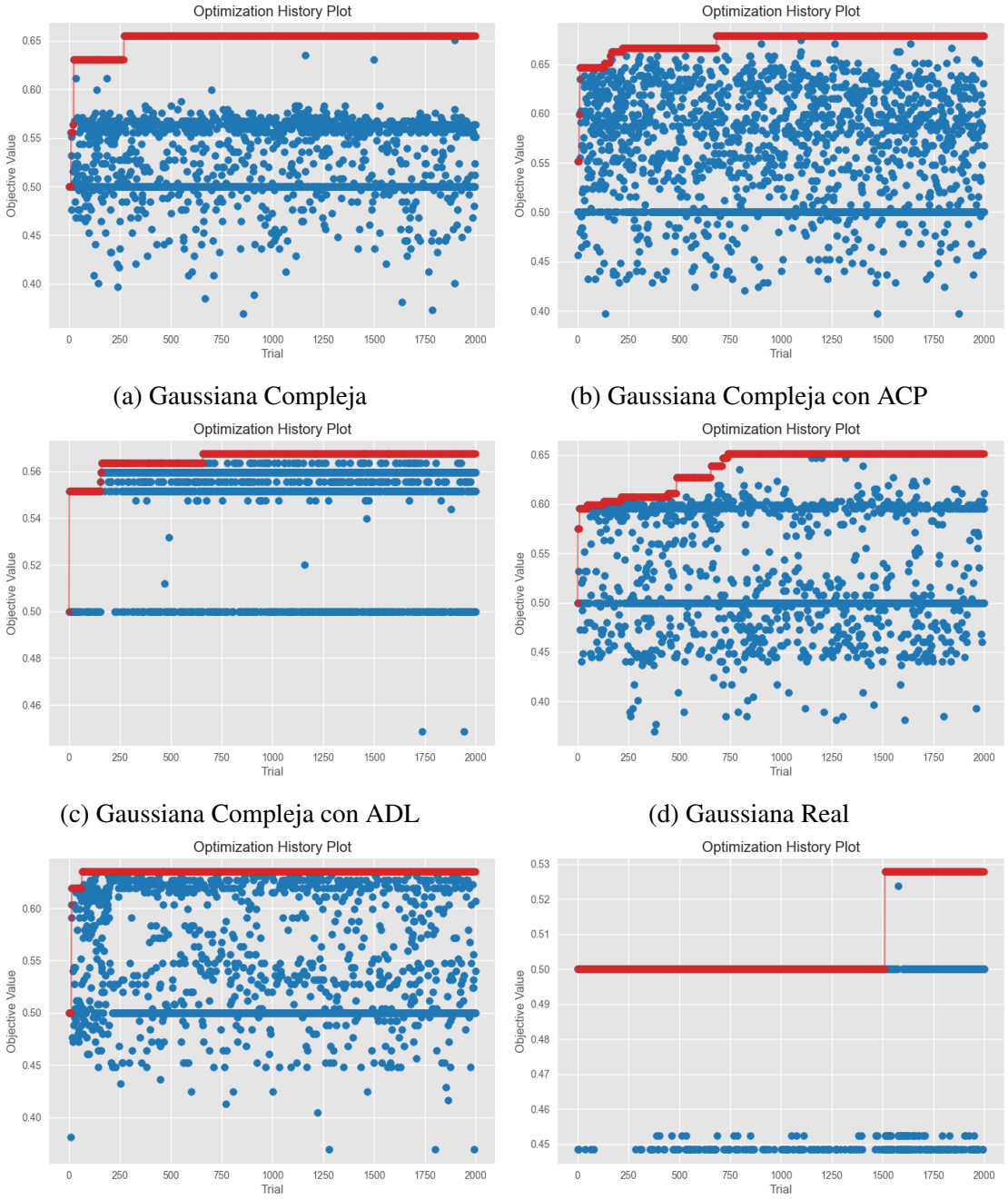
(c) Shannon con ADL

(d) Sombrero Mexicano

(e) Sombrero Mexicano con ACP

(f) Sombrero Mexicano con ADL

Figura C.6: Gráficas de optimización de hiperparámetros para Bosques Aleatorios con las wavelets Shannon y Sombrero Mexicano.



(a) Gaussiana Compleja

(b) Gaussiana Compleja con ACP

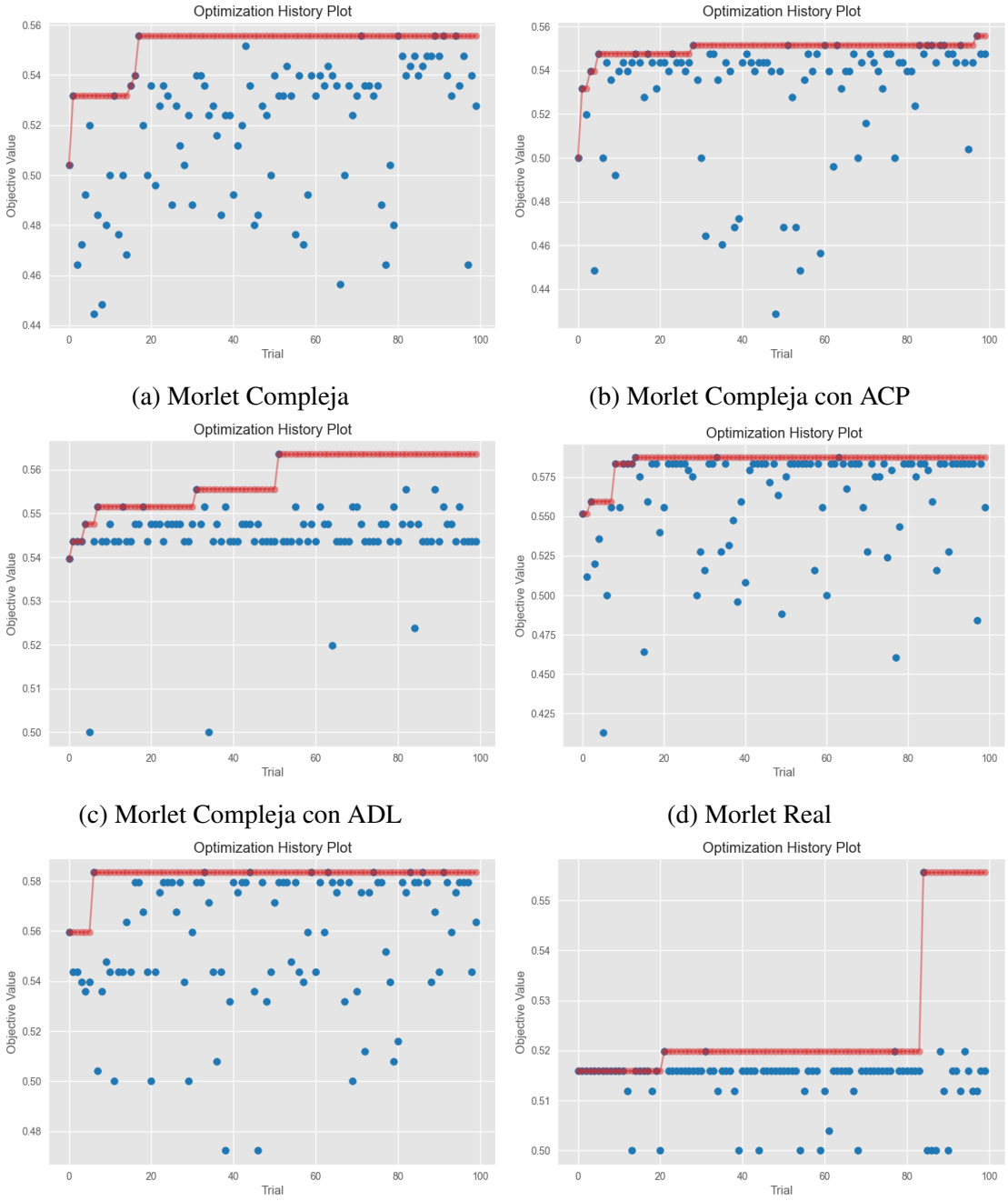
(c) Gaussiana Compleja con ADL

(d) Gaussiana Real

(e) Gaussiana Real con ACP

(f) Gaussiana Real con ADL

Figura C.7: Gráficas de optimización de hiperparámetros para Bosques Aleatorios con las wavelets Gaussiana Compleja y Gaussiana Real.



(a) Morlet Compleja

(b) Morlet Compleja con ACP

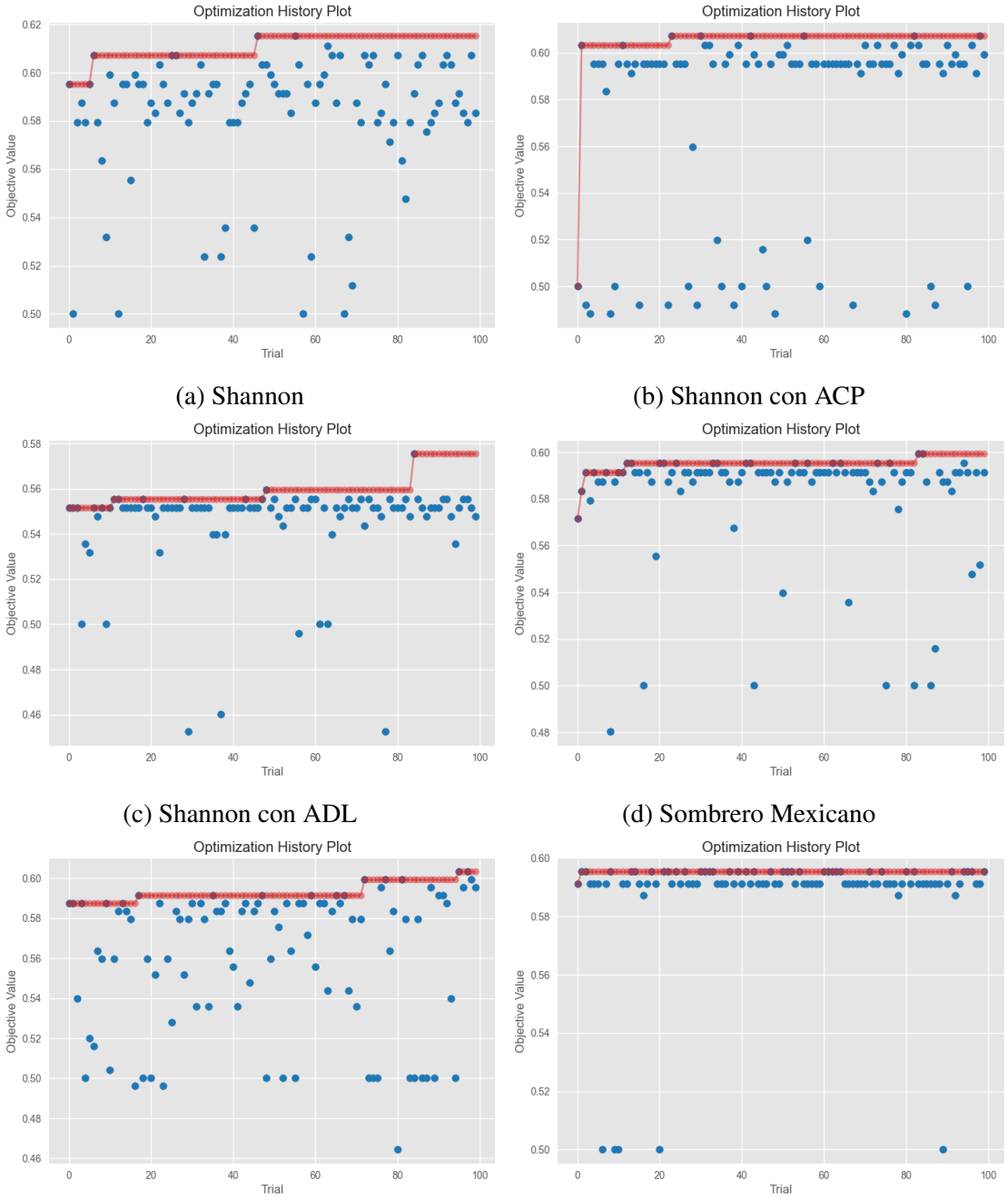
(c) Morlet Compleja con ADL

(d) Morlet Real

(e) Morlet Real con ACP

(f) Morlet Real con ADL

Figura C.8: Gráficas de optimización de hiperparámetros para Redes Neuronales con las wavelets Morlet Compleja y Morlet Real.



(a) Shannon

(b) Shannon con ACP

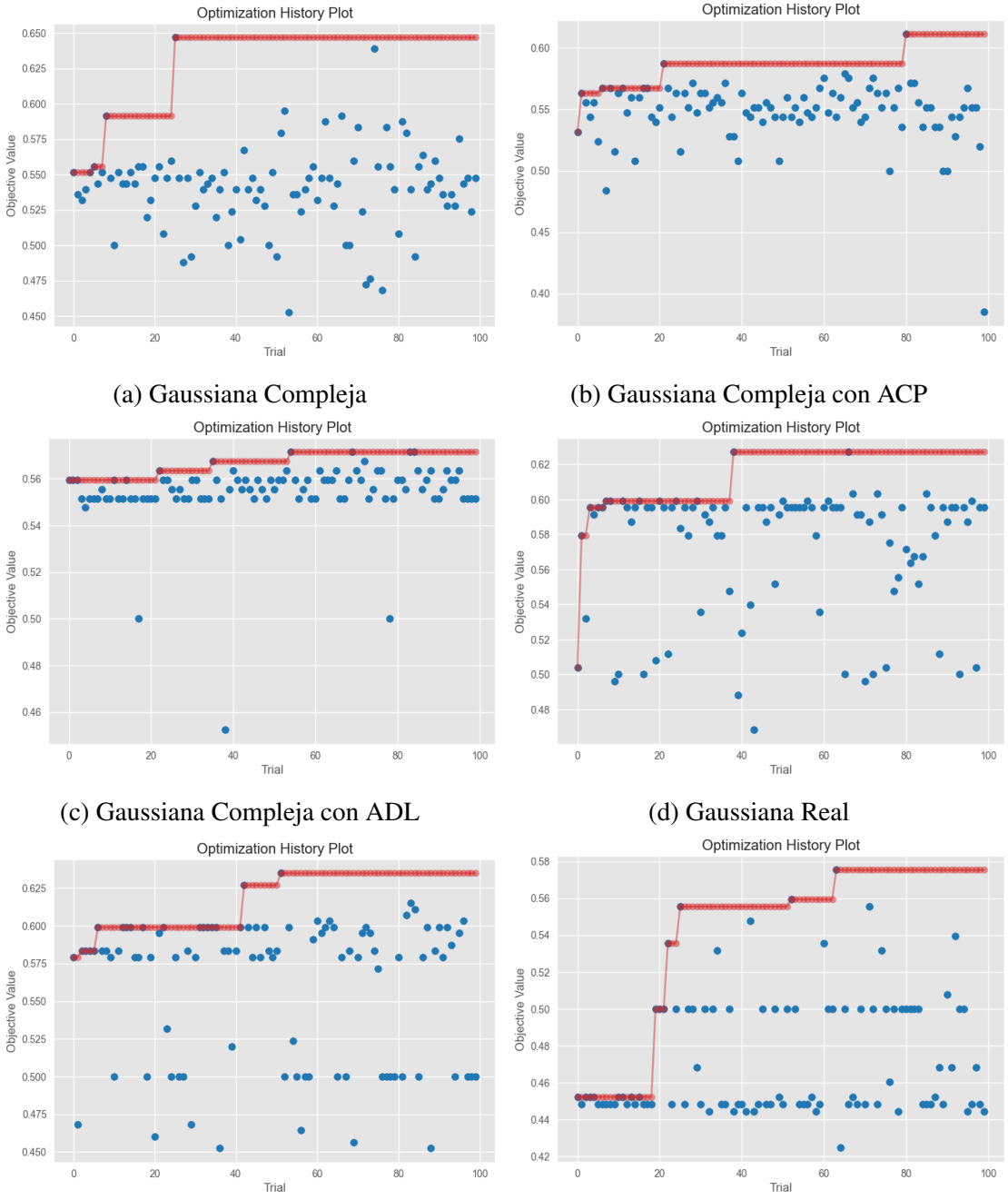
(c) Shannon con ADL

(d) Sombrero Mexicano

(e) Sombrero Mexicano con ACP

(f) Sombrero Mexicano con ADL

Figura C.9: Gráficas de optimización de hiperparámetros para Redes Neuronales con las wavelets Shannon y Sombrero Mexicano.



(a) Gaussiana Compleja

(b) Gaussiana Compleja con ACP

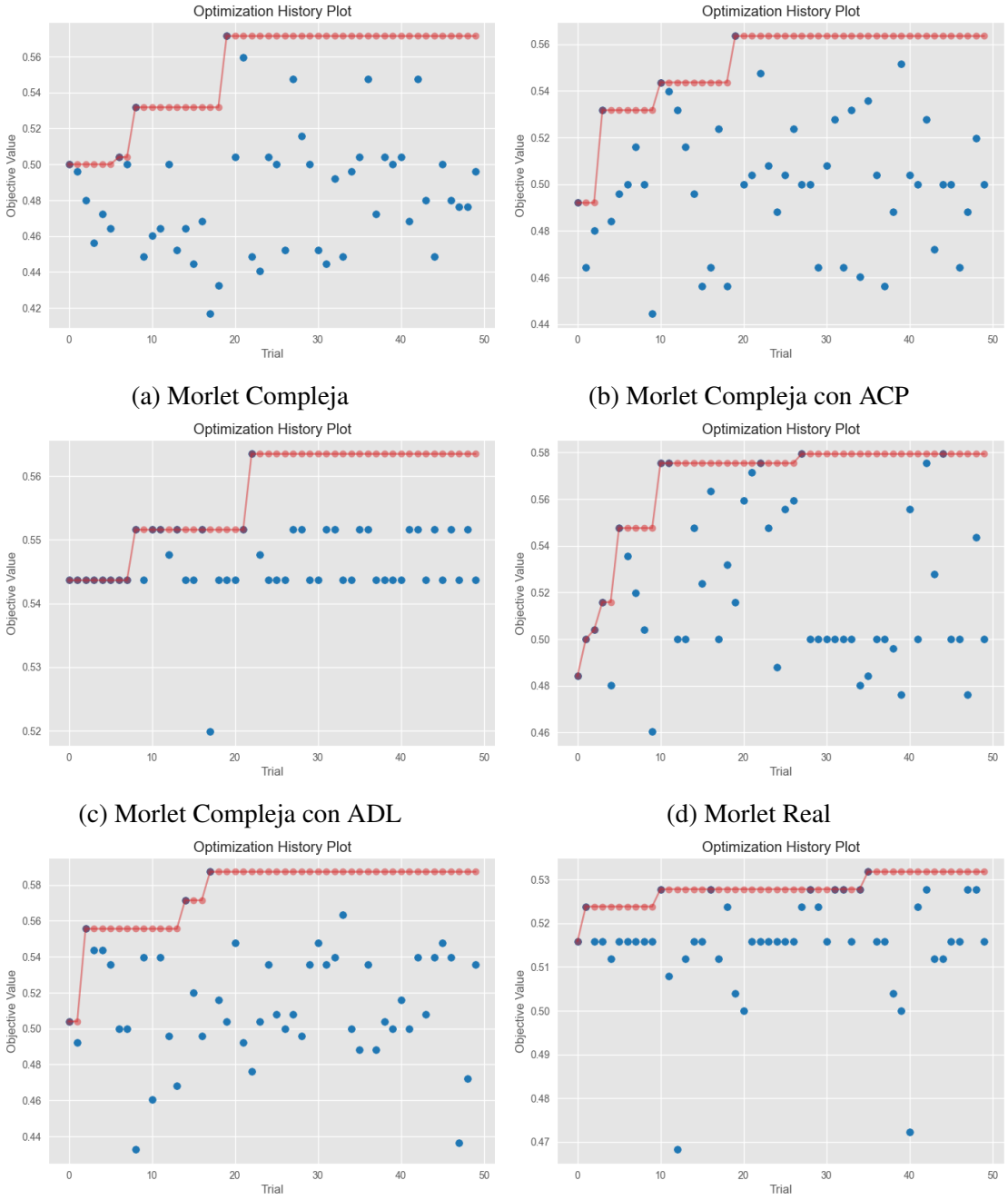
(c) Gaussiana Compleja con ADL

(d) Gaussiana Real

(e) Gaussiana Real con ACP

(f) Gaussiana Real con ADL

Figura C.10: Gráficas de optimización de hiperparámetros para Redes Neuronales con las wavelets Gaussiana Compleja y Gaussiana Real.



(a) Morlet Compleja

(b) Morlet Compleja con ACP

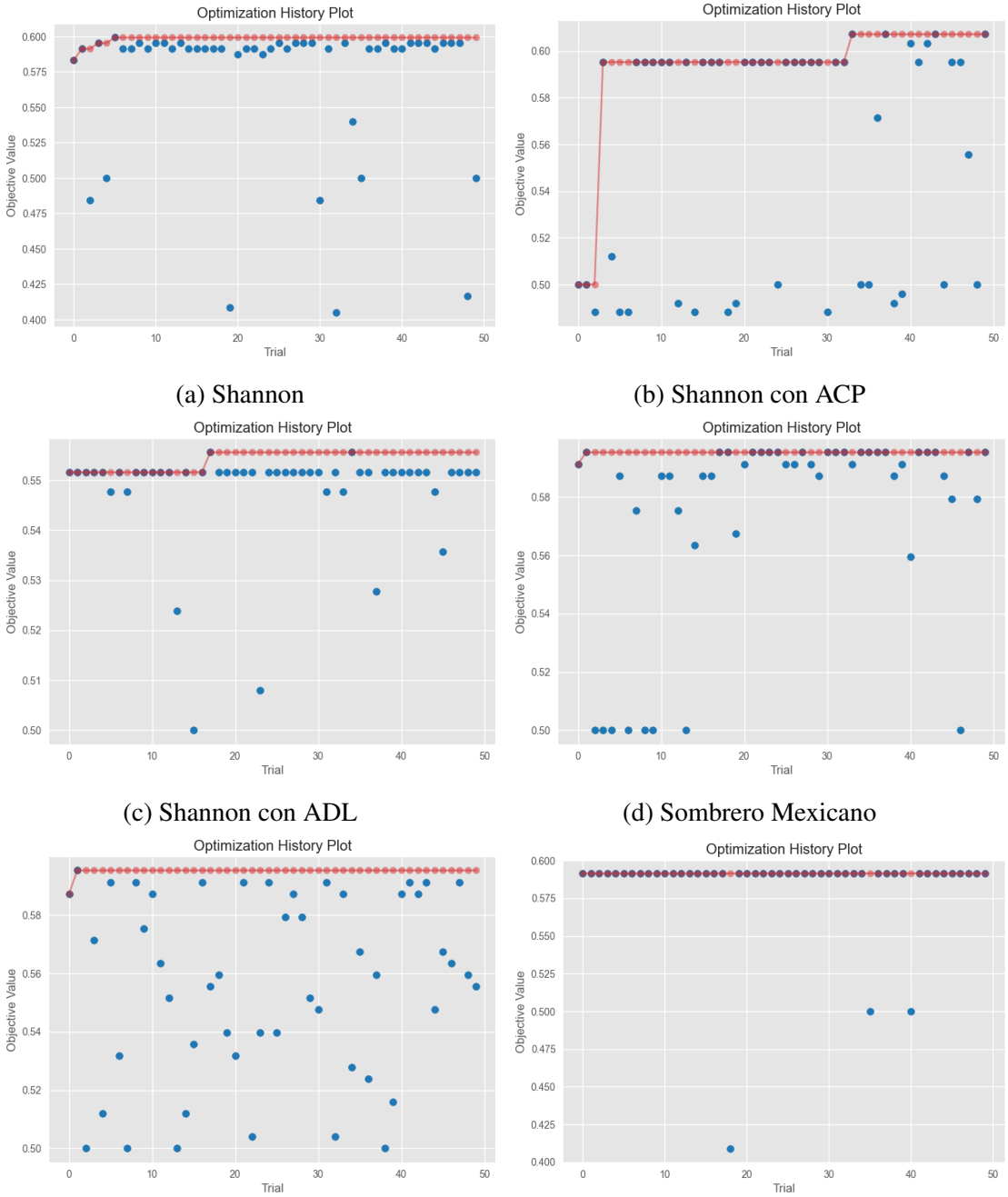
(c) Morlet Compleja con ADL

(d) Morlet Real

(e) Morlet Real con ACP

(f) Morlet Real con ADL

Figura C.11: Gráficas de optimización de hiperparámetros para Redes Neuronales Recurrentes con las wavelets Morlet Compleja y Morlet Real.



(a) Shannon

(b) Shannon con ACP

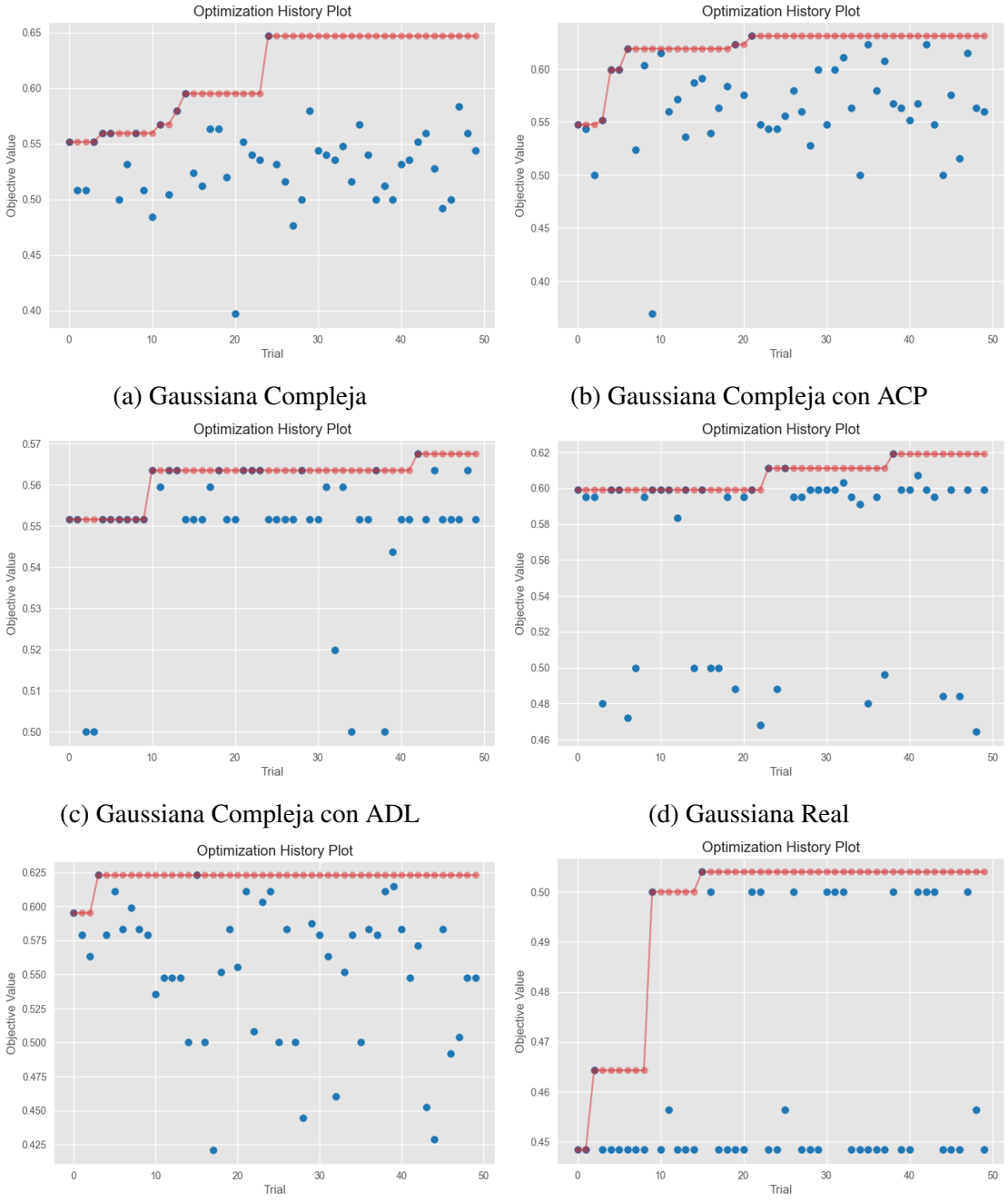
(c) Shannon con ADL

(d) Sombrero Mexicano

(e) Sombrero Mexicano con ACP

(f) Sombrero Mexicano con ADL

Figura C.12: Gráficas de optimización de hiperparámetros para Redes Neuronales Recurrentes con las wavelets Shannon y Sombrero Mexicano.



(a) Gaussiana Compleja

(b) Gaussiana Compleja con ACP

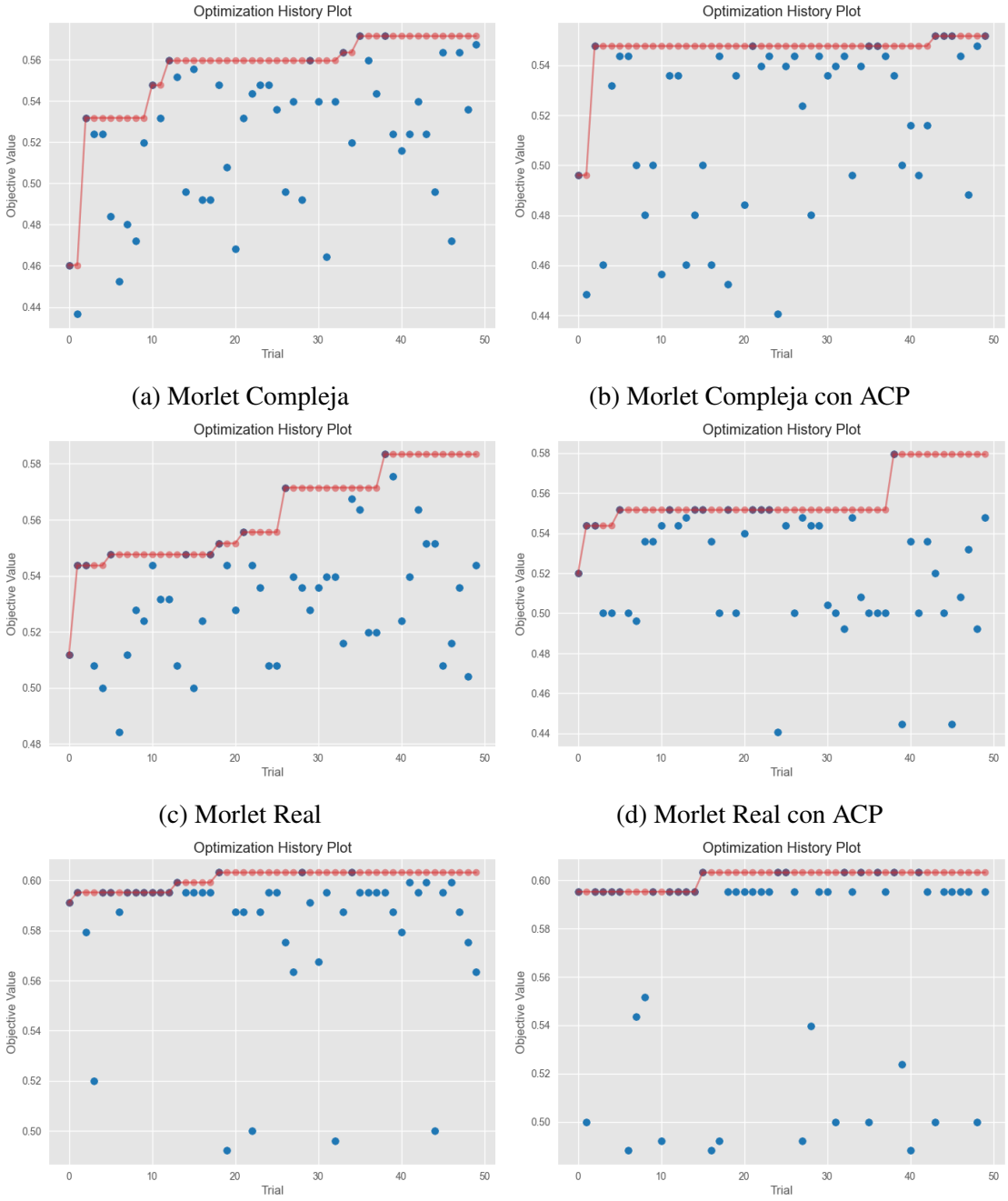
(c) Gaussiana Compleja con ADL

(d) Gaussiana Real

(e) Gaussiana Real con ACP

(f) Gaussiana Real con ADL

Figura C.13: Gráficas de optimización de hiperparámetros para Redes Neuronales Recurrentes con las wavelets Gaussiana Compleja y Gaussiana Real.



(a) Morlet Compleja

(b) Morlet Compleja con ACP

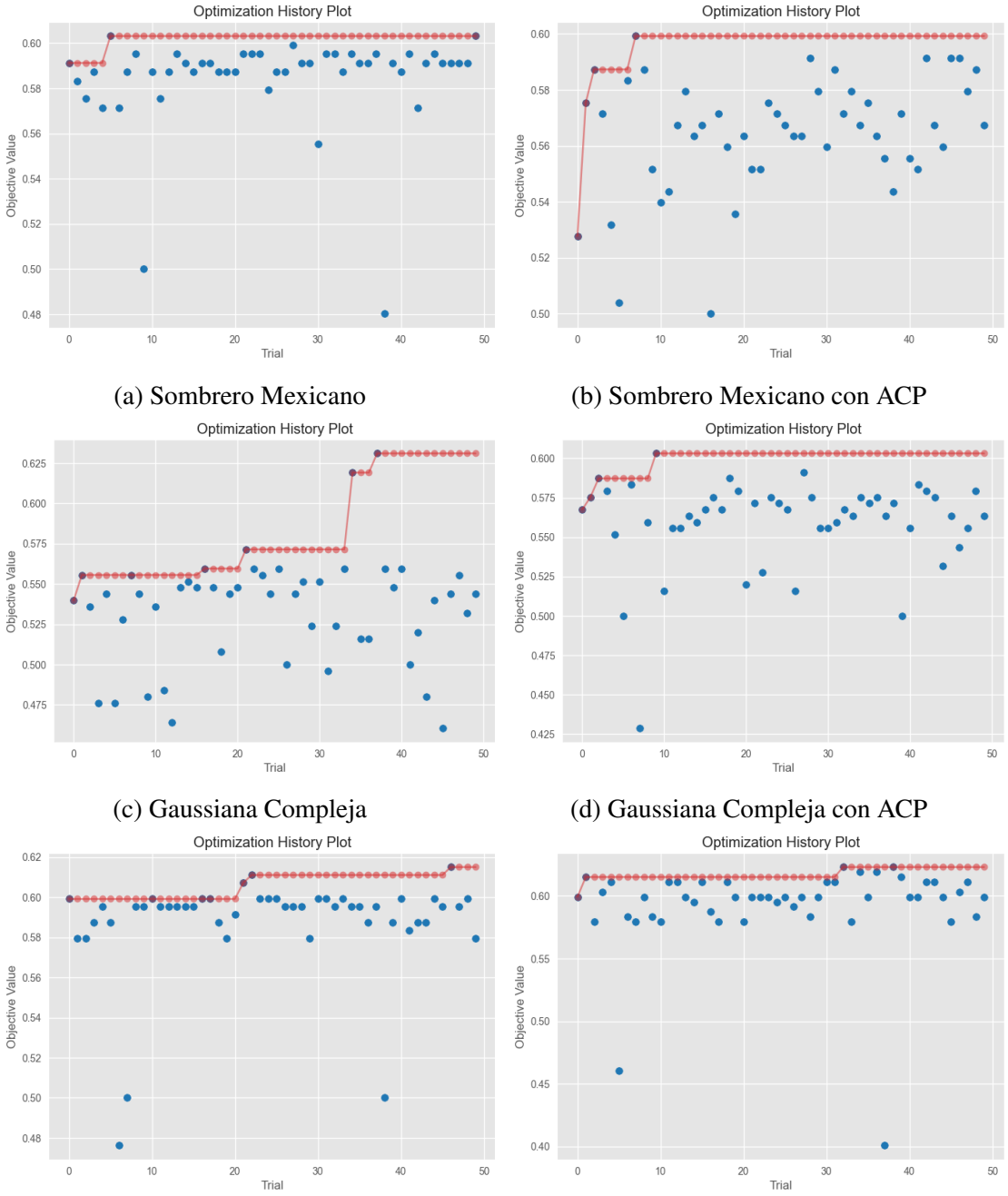
(c) Morlet Real

(d) Morlet Real con ACP

(e) Shannon

(f) Shannon con ACP

Figura C.14: Gráficas de optimización de hiperparámetros para Redes Neuronales Convolucionales con las wavelets Morlet Compleja, Morlet Real y Shannon.



(a) Sombrero Mexicano

(b) Sombrero Mexicano con ACP

(c) Gaussiana Compleja

(d) Gaussiana Compleja con ACP

(e) Gaussiana Real

(f) Gaussiana Real con ACP

Figura C.15: Gráficas de optimización de hiperparámetros para Redes Neuronales Convolucionales con las wavelets Sombrero Mexicano, Gaussiana Compleja y Gaussiana Real.