IDENTIFICACIÓN DE EXOPLANETAS USANDO MODELOS DE CLASIFICACIÓN

POR:

Juan José Gomez Mejia

MATERIA:

Introducción a la inteligencia artificial

PROFESOR:

Raul Ramos Pollan

UNIVERSIDAD DE ANTIOQUIA

FACULTAD DE INGENIERÍA

MEDELLÍN 2023

Contenido

1. Introducción

El descubrimiento de exoplanetas se ha convertido en una tarea muy importante y emocionante para la astronomía moderna; desde el descubrimiento del primer exoplaneta en 1995 por los astrónomos Michel Mayor y Didier Queloz, al que llamaron 51 Pegasi b, utilizando la técnica de velocidad radial, que es una técnica de detección indirecta que se basa en el análisis del efecto gravitacional de un planeta en su estrella anfitriona, realizando así la primera detección directa del espectro de luz visible reflejado por un exoplaneta; hasta la reciente detección espectral de transmisión atmosférica del exoplaneta WASP-39b capturado por el espectrógrafo de infrarrojo cercano del telescopio James Webb, revelando la primera evidencia clara de dióxido de carbono en la atmósfera de un planeta fuera del Sistema Solar.

Porque resulta que detectar exoplanetas, e incluso intentar obtener datos en espectro de luz visible es extremadamente difícil, debido a varios desafíos técnicos y astronómicos involucrados:

En primer lugar, los exoplanetas son objetos muy pequeños y débiles en comparación con sus estrellas anfitrionas. Debido a que los exoplanetas no emiten su propia luz, sino que reflejan la luz de su estrella, cualquier señal de un exoplaneta es enmascarada por el brillo abrumador de su estrella. Esto hace que sea difícil separar la señal del exoplaneta de la señal de la estrella, especialmente cuando el exoplaneta está muy cerca de su estrella anfitriona.
En segundo lugar, los exoplanetas están muy lejos, lo que hace que incluso los telescopios más potentes tengan dificultades para obtener datos detalladas de ellos. Para hacerlo, los telescopios deben ser capaces de detectar la luz de un exoplaneta que está separado por solo unos pocos píxeles de su estrella anfitriona, lo que requiere técnicas avanzadas de procesamiento de imágenes y una gran estabilidad en la observación.

Sin embargo, actualmente existe una grande lista de exoplanetas confirmados, siendo más de 5000, ubicados en aproximadamente más de 3650 sistemas planetarios, descubiertos entre las distintas misiones espaciales como Kepler, TESS, CHEOPS, y técnicas de detección como los tránsitos planetarios, velocidad radial, microlentes gravitatorias y técnicas de imagen directa.

2. Planteamiento del problema

La inteligencia artificial es una herramienta poderosa que puede generar valor a partir de los datos en diversos campos como en la astronomía. Por lo que, en este proyecto, nos enfocaremos en su aplicación para clasificar exoplanetas utilizando algoritmos de Machine Learning. El objetivo es analizar el conjunto de datos proporcionado por la **NASA**, llamado "**Kepler Exoplanet Search Results**", que contiene información sobre la búsqueda de exoplanetas realizada por la misión Kepler.

El dataset proporcionado incluye una amplia variedad de características de las estrellas y los posibles exoplanetas detectados. Algunas de estas características incluyen parámetros físicos como su temperatura, luminosidad y radio, así como información sobre las señales y tránsitos que podrían indicar la presencia de un exoplaneta; de ésta manera, la clasificación precisa de los exoplanetas es de gran importancia en la astronomía, ya que nos permite comprender mejor la formación y evolución de los sistemas planetarios.
Además, estos modelos pueden ser utilizados para identificar objetivos potenciales para futuras misiones espaciales y estudios más detallados, así como para seleccionar y priorizar las observaciones y recursos disponibles.

Así al implementar algoritmos de Machine Learning en el campo de la astrofísica y la exploración espacial, se abre un amplio abanico de posibilidades para la automatización y aceleración del análisis de datos, así como para el descubrimiento de nuevos conocimientos en el universo.

Este proyecto contribuirá al desarrollo de técnicas y herramientas que pueden ser aplicadas en futuras investigaciones astronómicas, avanzando en nuestro entendimiento de los exoplanetas y su diversidad. Por esta razón, el objetivo principal de este proyecto es desarrollar un modelo de Machine Learning capaz de clasificar correctamente los exoplanetas en base a las características proporcionadas. Se explorarán diferentes algoritmos de clasificación, como Random Forest y Redes Neuronales, con el fin de encontrar resultados relevantes para el estudio.

2.1. Dataset

Los datos vienen del sitio web Kaggle, con nombre **Kepler Exoplanet Search Results**, proporcionados por la NASA; son datos obtenidos por la misión espacial Kepler, capturados por el telescopio espacial Kepler lanzado en 2009, y operando hasta 2018, siendo la punta de la lanza en detección de exoplanetas, donde más de los cerca de 5000 candidatos planetarios encontrados hasta la fecha, más de 3.200 ahora han sido verificados, y 2.325 de estos fueron descubiertos por el Kepler.
De esta forma, el dataset se conforma de 50 columnas o características y 9564 filas o muestras para poder predecir si un exoplaneta **detectado**, puede clasificarse en:

CANDIDATO: Se ha detectado un tránsito en los datos del telescopio, pero no se han realizado observaciones adicionales para confirmar si el tránsito es causado por un verdadero planeta.

CONFIRMADO: Se ha verificado mediante observaciones adicionales que el tránsito del candidato a exoplaneta es causado por un planeta real que orbita alrededor de una estrella.

FALSO POSITIVO: Se ha determinado que el tránsito del candidato a exoplaneta es causado por una señal falsa, como una estrella de fondo, un artefacto o una perturbación en los datos.

Teniendo así un problema de clasificación múltiple.

¡NOTA!
*De acuerdo a la cantidad de muestras que son **FALSO POSITIVO** respecto a las demás, puede que se tenga que eliminar y trabajar sólo con* **CANDIDATO** y **CONFIRMADO;** más adelante se hablará sobre la decisión establecida.

Por otra parte, entre las características más importantes están:

- ***koi_disposition***: El estado de la KOI en función de la validación de su existencia. Puede ser "CANDIDATE", "CONFIRMED", o "FALSE POSITIVE". ***(La variable objetivo del dataset)***
- ***kepid***: El ID de Kepler para el objetivo observado.
- ***kepoi_name***: El nombre del planeta Kepler Object of Interest (KOI) asignado por el equipo de Kepler.
- ***koi_pdisposition:*** La probabilidad de que el planeta tenga una disposición positiva (confirmado) según un modelo estadístico. ***(Otra posible salida del dataset)***

¡NOTA!
Como se tienen 2 salidas, se excluirá esta columna y se trabajará con ***koi_disposition****. A fin de cuentas,* ***koi_pdisposition*** *sólo es una columna de decisión provisional sobre la existencia de exoplanetas.*

- ***koi_score***: La confiabilidad de la detección de un tránsito planetario en un sistema estelar.
- ***koi_period***: El período orbital del planeta, en días.
- ***koi_time0bk***: El tiempo de tránsito del primer tránsito, en días julianos.
- ***koi_duration***: La duración del tránsito, en horas.
- ***koi_prad***: El radio del planeta, en radios terrestres.
- ***koi_teq***: La temperatura de equilibrio del planeta, en grados Kelvin.
- ***koi_insol***: La insolación recibida por el planeta, en unidades de la Tierra.
- ***koi_steff***: La temperatura efectiva de la estrella, en Kelvin.
- ***koi_slogg***: La gravedad superficial de la estrella, en cm/s^2.
- ***koi_srad***: El radio de la estrella, en radios solares.
- ***koi_kepmag***: La magnitud aparente en banda Kepler del objetivo observado.
- ***koi_depth***: La profundidad del tránsito, en partes por millón (ppm).
- ***koi_count:*** El número de planetas en el sistema estelar.
- ***koi_period_err1:*** El error positivo en el período orbital del planeta, en días.
- ***koi_period_err2***: El error negativo en el período orbital del planeta, en días.
- ***koi_impact***: Parámetro de impacto estelar, que mide la distancia mínima entre el centro de la estrella y el centro del planeta en el momento del tránsito, en unidades de radio estelar.
- ***koi_smet:*** Metalicidad estelar, que mide la cantidad de elementos más pesados que el helio, presentes en la estrella anfitriona, en unidades de logaritmo de la relación con el Sol.
- ***koi_srho:*** Densidad estelar, que mide la densidad de la estrella anfitriona, en unidades de g/cm^3.
- ***koi_tce_plnt_num***: Número de planeta en el Sistema Multiplanet Transiting Candidate (MTC) al que pertenece la KOI.
- ***koi_quarters***: El trimestre en que se observó el objetivo.
- ***koi_disposition_score***: La puntuación de confianza asignada a la KOI por el equipo de validación de Kepler.
- ***koi_fpflag_nt:*** El número de veces que la KOI cruzó el borde de la máscara de destino.
- ***koi_fpflag_ss***: La cantidad de detecciones estadísticamente significativas de tránsito secundario.

2.2. Métricas

Para clasificar exoplanetas detectados como **candidatos**, **confirmados** o **falsos positivos**, es importante seleccionar las métricas adecuadas para evaluar el rendimiento del modelo.

En este caso, como se trataría de un problema de clasificación, se pueden utilizar métricas como el **Accuracy**, la **Precisión**, el **Recall**, el **F1 Score,** o la curva **ROC-AUC**; donde cada métrica se describe en función de los resultados dados en la llamada **matriz de confusión**:
La matriz de confusión permite visualizar la tasa de aciertos y errores de un modelo de clasificación. Es una tabla que muestra el número de casos clasificados como verdaderos positivos, falsos positivos, verdaderos negativos y falsos negativos. Se representa de la siguiente manera:

		Predicción	
		Positivos	**Negativos**
Observación	**Positivos**	Verdaderos Positivos (VP)	Falsos Negativos (FN)
	Negativos	Falsos Positivos (FP)	Verdaderos Negativos (VN)

Donde:
TP (true positives): número de casos positivos que fueron correctamente identificados por el modelo.
TN (true negatives): número de casos negativos que fueron correctamente identificados por el modelo.
FP (false positives): número de casos negativos que fueron incorrectamente clasificados como positivos por el modelo.
FN (false negatives): número de casos positivos que fueron incorrectamente clasificados como negativos por el modelo.

Por lo tanto, **la métrica Accuracy** mide la proporción de predicciones correctas que un modelo realiza sobre el total de predicciones realizadas. Es decir, es la cantidad de casos clasificados correctamente como positivos o negativos en relación al total de casos. La fórmula matemática es la siguiente:

$$Accuraccy = \frac{TP + TN}{TP + FP + TN + FN}$$

La métrica de Precisión mide la proporción de casos positivos que fueron correctamente identificados como tales por el modelo en relación al total de casos clasificados como positivos. La fórmula para calcular la precisión es la siguiente:

$$Precision = \frac{TP}{TP + FP}$$

Por otra parte, **la métrica Recall** (o tasa de verdaderos positivos, TPR) es la métrica de evaluación también utilizada en clasificación que mide la capacidad del modelo para identificar correctamente las instancias positivas en relación al total de instancias positivas en el conjunto de datos. La fórmula para calcular el recall es la siguiente:

$$Recall = \frac{TP}{TP + FN}$$

El **F1 Score** es una medida de la precisión del modelo que considera tanto la precisión como el recall, siendo una medida balanceada que busca obtener un equilibrio entre ambas métricas. La fórmula matemática es la siguiente:

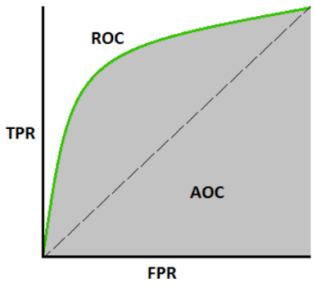
$$F1\ Score = \frac{2(Precision * Recall)}{Precision + Recall}$$

Y finalmente se podría utilizar **La métrica AUC** (Área Bajo la Curva) se utiliza para evaluar la capacidad del modelo para distinguir entre dos clases: positiva y negativa. Un valor de AUC cercano a 0,5 indica un modelo que clasifica al azar, mientras que un valor cercano a 1 indica un modelo muy preciso. La fórmula para el cálculo del AUC es la siguiente:

$$AUC = \int TPR\ d(FPR)$$

$$TPR = \frac{TP}{TP + FN}$$

$$FPR = \frac{FP}{FP + TN}$$



Donde TPR es la tasa de verdaderos positivos y FPR es la tasa de falsos positivos. La integral se realiza sobre el rango completo de FPR. La curva ROC se utiliza para obtener los valores de TPR y FPR para diferentes umbrales de clasificación, y se traza en un gráfico con TPR en el eje y y FPR en el eje x. El AUC se calcula a partir de esta curva, y proporciona una medida de la capacidad del modelo para distinguir entre las dos clases; por lo tanto, en nuestro caso, un valor alto de AUC indicaría que el modelo es capaz de distinguir entre las dos clases con alta precisión.

3. Naturaleza del Dataset

Para la clasificación de exoplanetas usando el dataset "**Kepler Exoplanet Search Results**.", primero es necesario entender la naturaleza del dataset y luego hacer un preprocesamiento antes de hacer cualquier entrenamiento; el análisis de los datos recopilados por los telescopios espaciales puede ser un desafío debido a la complejidad, el ruido que presentan, así como la recopilación e interpretación de los mismos. Es por eso que el preprocesamiento de los datos es esencial si se quieren obtener resultados más precisos y confiables en la clasificación de exoplanetas.

En la ***Figura 1*** se muestra la tabla del contenido del dataset; a grandes rasgos podemos observar variables categóricas, y continuas, así como nombres, numeraciones y algunos datos faltantes. Además, podemos considerar que el nombramiento de columnas puede no representar o dar a entender lo que significan en cuestión.

	rowid	kepid	kepoi_name	kepler_name	koi_disposition	koi_pdisposition	koi_score	koi_fpflag_nt	koi_fpflag_ss	koi_fpflag_co	...	ra	dec	koi_kepmag
0	1	10797460	K00752.01	Kepler-227 b	CONFIRMED	CANDIDATE	1.000	0	0	0	...	291.93423	48.141651	15.347
1	2	10797460	K00752.02	Kepler-227 c	CONFIRMED	CANDIDATE	0.969	0	0	0	...	291.93423	48.141651	15.347
2	3	10811496	K00753.01	NaN	FALSE POSITIVE	FALSE POSITIVE	0.000	0	1	0	...	297.00482	48.134129	15.436
3	4	10848459	K00754.01	NaN	FALSE POSITIVE	FALSE POSITIVE	0.000	0	1	0	...	285.53461	48.285210	15.597
4	5	10854555	K00755.01	Kepler-664 b	CONFIRMED	CANDIDATE	1.000	0	0	0	...	288.75488	48.226200	15.509
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
9559	9560	10031643	K07984.01	NaN	FALSE POSITIVE	FALSE POSITIVE	0.000	0	0	0	...	298.74921	46.973351	14.478
9560	9561	10090151	K07985.01	NaN	FALSE POSITIVE	FALSE POSITIVE	0.000	0	1	1	...	297.18875	47.093819	14.082
9561	9562	10128825	K07986.01	NaN	CANDIDATE	CANDIDATE	0.497	0	0	0	...	286.50937	47.163219	14.757
9562	9563	10147276	K07987.01	NaN	FALSE POSITIVE	FALSE POSITIVE	0.021	0	0	1	...	294.16489	47.176281	15.385
9563	9564	10156110	K07989.01	NaN	FALSE POSITIVE	FALSE POSITIVE	0.000	0	0	1	...	297.00977	47.121021	14.826

9564 rows × 50 columns

*Figura 1 – Dataset **Kepler Exoplanet Search Results.***

Por lo tanto, es recomendable hacer un renombrado de columnas, para tener un mejor contexto de lo que significa cada característica.
La ***Figura 2*** muestra un renombrado de columnas más adecuado:

rowid	rowid
kepid	KepID
kepoi_name	KOIName
kepler_name	KeplerName
koi_disposition	ExoplanetArchiveDisposition
koi_pdisposition	DispositionUsingKeplerData
koi_score	DispositionScore
koi_fpflag_nt	NotTransit-LikeFalsePositiveFlag
koi_fpflag_ss	koi_fpflag_ss
koi_fpflag_co	CentroidOffsetFalsePositiveFlag
koi_fpflag_ec	EphemerisMatchIndicatesContaminationFalsePositiveFlag
koi_period	OrbitalPeriod_days
koi_period_err1	OrbitalPeriodUpperUnc_days
koi_period_err2	OrbitalPeriodLowerUnc_days
koi_time0bk	TransitEpoch_BKJD
koi_time0bk_err1	TransitEpochUpperUnc_BKJD
koi_time0bk_err2	TransitEpochLowerUnc_BKJD
koi_impact	ImpactParamete
koi_impact_err1	ImpactParameterUpperUnc
koi_impact_err2	ImpactParameterLowerUnc
koi_duration	TransitDuration_hrs
koi_duration_err1	TransitDurationUpperUnc_hrs
koi_duration_err2	TransitDurationLowerUnc_hrs
koi_depth	TransitDepth_ppm
koi_depth_err1	TransitDepthUpperUnc_ppm
koi_depth_err2	TransitDepthLowerUnc_ppm
koi_prad	PlanetaryRadius_Earthradii
koi_prad_err1	PlanetaryRadiusUpperUnc_Earthradii
koi_prad_err2	PlanetaryRadiusLowerUnc_Earthradii
koi_teq	EquilibriumTemperatureK
koi_teq_err1	EquilibriumTemperatureUpperUncK
koi_teq_err2	EquilibriumTemperatureLowerUncK
koi_insol	InsolationFlux_Earthflux
koi_insol_err1	InsolationFluxUpperUnc_Earthflux
koi_insol_err2	InsolationFluxLowerUnc_Earthflux
koi_model_snr	TransitSignal-to-Nois
koi_tce_plnt_num	TCEPlanetNumbe
koi_tce_delivname	TCEDeliver
koi_steff	StellarEffectiveTemperatureK
koi_steff_err1	StellarEffectiveTemperatureUpperUncK
koi_steff_err2	StellarEffectiveTemperatureLowerUncK
koi_slogg	StellarSurfaceGravity_log10(cm/s**2)
koi_slogg_err1	StellarSurfaceGravityUpperUnc_log10(cm/s**2)
koi_slogg_err2	StellarSurfaceGravityLowerUnc_log10(cm/s**2)
koi_srad	StellarRadius_Solarradii
koi_srad_err1	StellarRadiusUpperUnc_Solarradii
koi_srad_err2	StellarRadiusLowerUnc_Solarradii
ra	RA_decimaldegrees
dec	Dec_decimaldegrees
koi_kepmag	Kepler-band_mag

Figura 2 - Renombrado de columnas.

3.1. Variable Objetivo

Como ya se mencionó anteriormente, la variable objetivo que se desea predecir es ***ExoplanetArchiveDisposition***; así**,** en la ***Figura 3*** podemos observar que la cantidad de **FALSE POSITIVE** es mayor respecto a **CANDIDATE** y **CONFIRMED**; esto podría alterar considerablemente cualquier tipo de resultado que se quiera obtener al haber un desbalance evidente, lo que significa que como decisión se decide trabajar sólo con **CANDIDATE** y **CONFIRMED**, tendríamos un problema biclase, y el tamaño del dataset se reduce a **(4541, 50).**

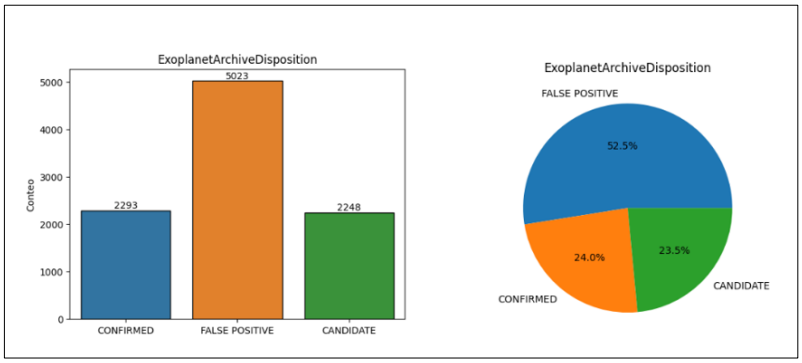


Figura 3 - Variable objetivo con 3 salidas.

En la ***Figura 4***, podemos observar que hay un mejor balance de clases.

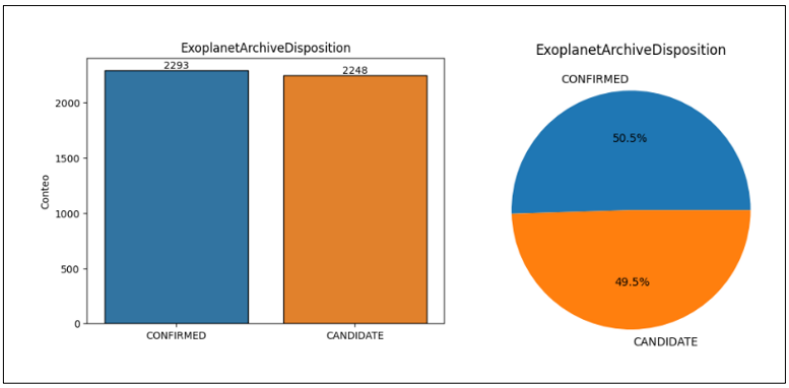


Figura 4 - Variable objetivo como un problema biclase

4. Tratamiento de datos

- **Eliminación de variables innecesarias:**
  Es necesario eliminar características irrelevantes como el ***rowid***, la ***KepID***, las columnas que sólo son nombres como la ***Kepler_name*** y las columnas que están completamente vacías como ***EquilibriumTemperatureUpperUncK*.**

- **Transformación de variables categóricas a categórico-numéricas:**
  Las variables categóricas no pueden ser usadas en el entrenamiento de un algoritmo. Por ello, hay que utilizar *One_Hot_Encoding* para ordenar algunas columnas como valores categórico-numéricos, en nuestro caso hay que hacerlo para la ***TCEDeliver*.**

4.1. Datos faltantes

Algo que también es importante analizar antes de entrenar un algoritmo de machine learning, es buscar datos faltantes en el dataset. En la ***Figura 5*** se muestra la cantidad de datos faltantes de las variables que poseen valores nulos o NaN.

Por lo tanto, hay que aplicar técnicas de limpieza y transformación de los datos para reparar los valores nulos; en nuestro caso, podemos utilizar la ***moda*** para datos numérico-categóricos y la ***media*** para datos continuos.

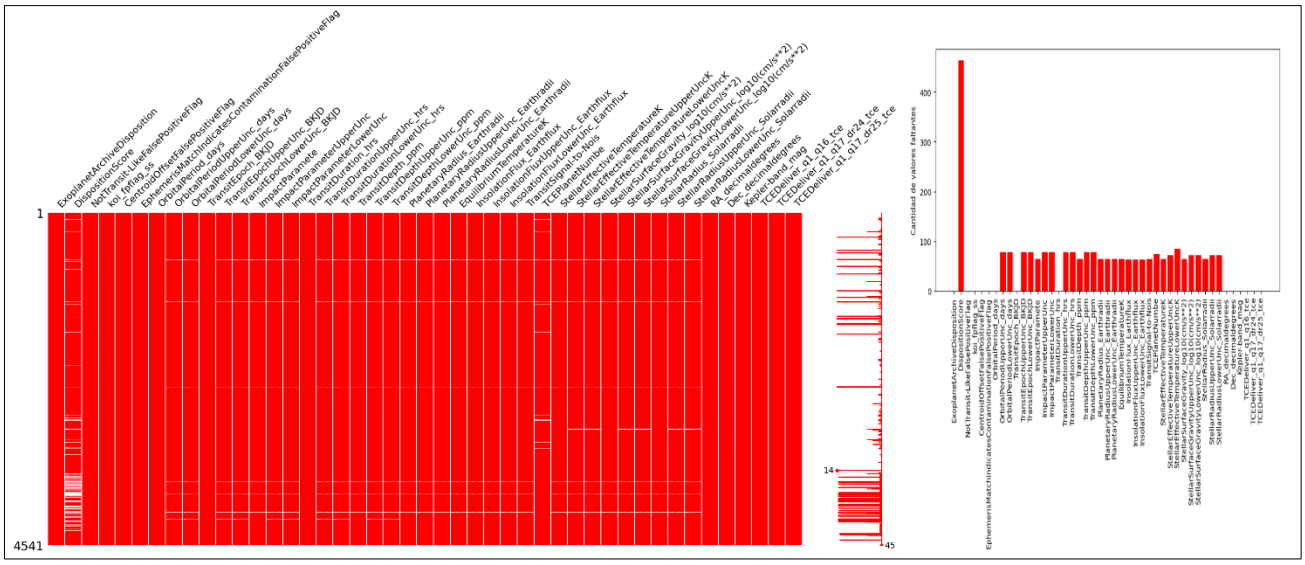


Figura 5 - Distribución y cantidad de datos faltantes

La ***Figura 6*** muestra cómo ahora todos los valores están corregidos.

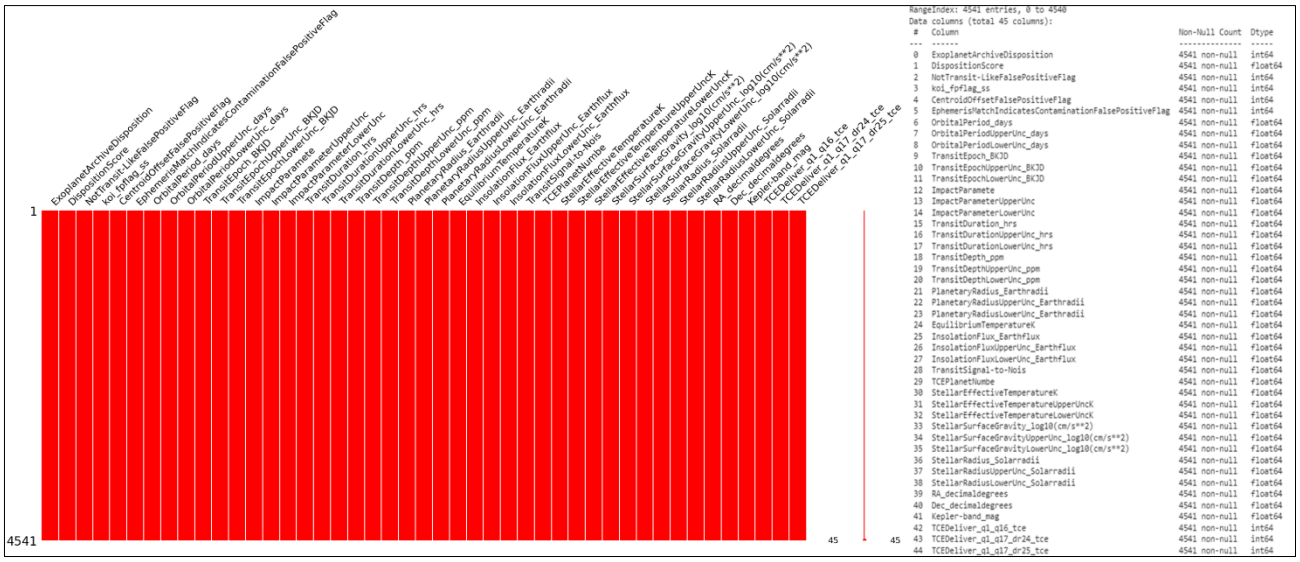


Figura 6 - Valores corregidos

5. Exploración de variables

Otra parte fundamental para entender la naturaleza del dataset, es hacer una exploración de variables, como analizar la correlación de todas las características entre sí; así como observar la distribución y frecuencia de los datos dentro del dataset.

5.1. Correlación de variables

La ***Figura 7*** muestra los valores de correlación que existen entre todas las características; así como con la variable objetivo. Se puede observar que ***TCEDeliver_q1_q17_dr24_tce*** y ***TransitDurationLowerUnc_hrs*** son las variables que tiene la mayor correlación. También se puede observar que en general, las variables ***OrbitalPeriodUpperUnc_days*** tienen una correlación tan baja que puede decirse que no están para nada relacionadas con ***ExoplanetArchiveDisposition***.

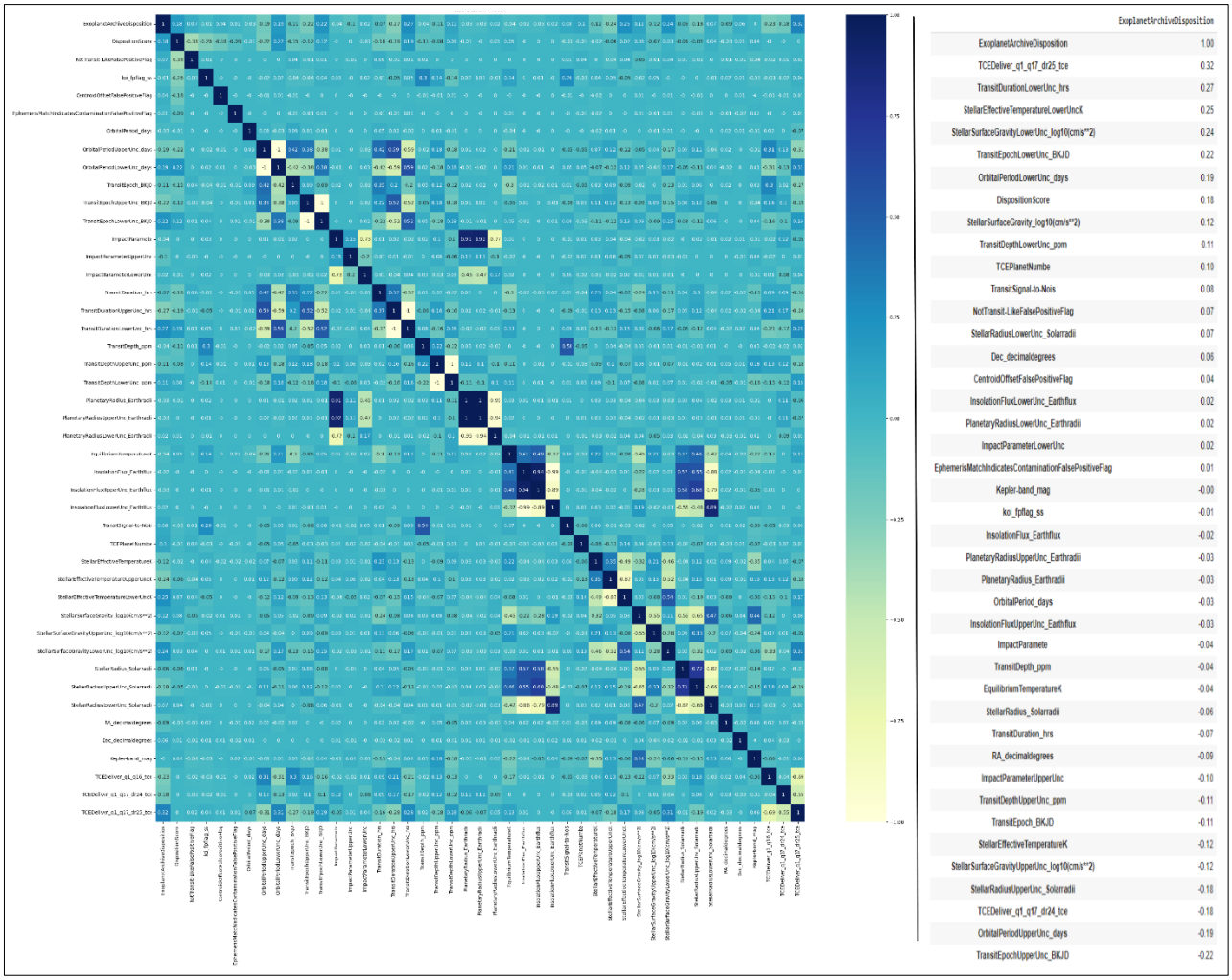


Figura 7 - Correlación de todas las variables

5.2. Distribución de las variables numéricas

En la ***Figura 8*** se muestran las frecuencias de cada variable. Se puede observar que algunas variables presentan una distribución semejante a la normal. Por otra parte, se destaca que hay variables con asimetría lateralmente marcada.

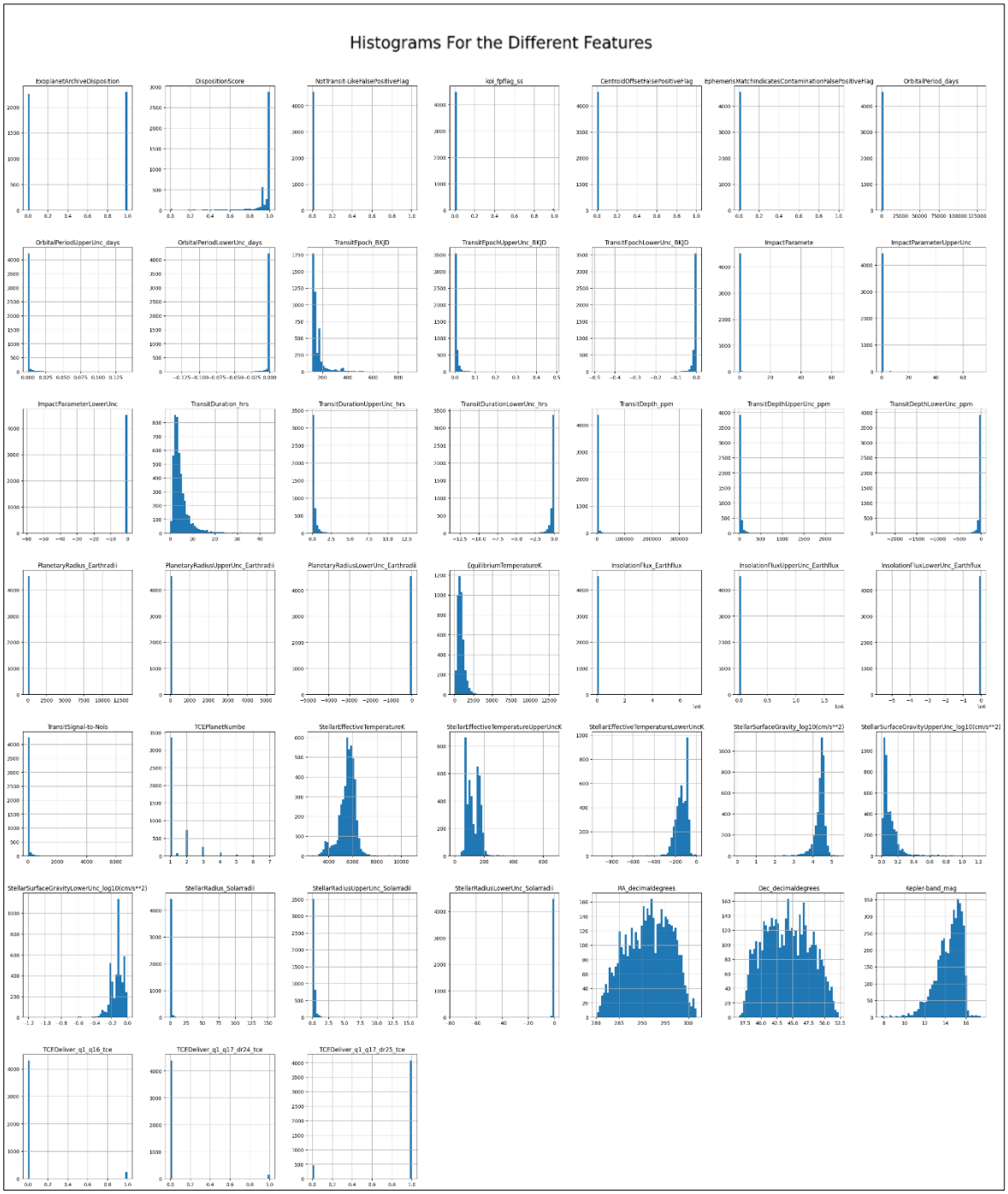


Figura 8 - Frecuencia de las variables.

Y en la ***Figura 9*** se muestra la distribución de los datos de cada característica en el dataset.

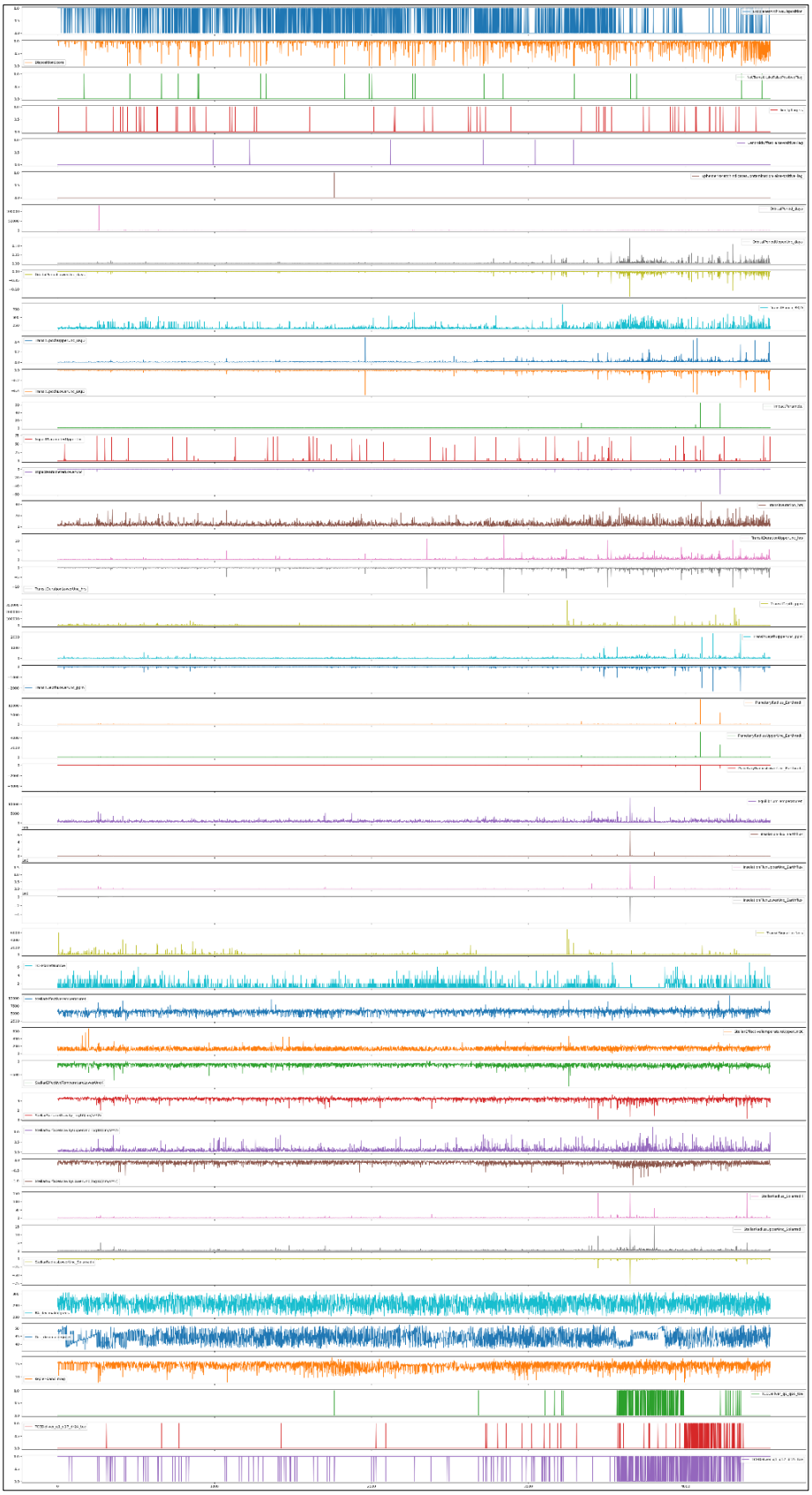


Figura 10 - Distribución de los datos alrededor de todo el dataset.

6. Algoritmos supervisados

6.1.1. Normalización de los datos

Aplicar una normalización a los datos puede ser conveniente para asegurar que las características estén en una escala comparable y evitar problemas relacionados con la escala y la distribución de los datos, de esta manera, utilizaremos una normalización conocida como estandarización **Z-score**. En este caso específico, se realizó el proceso de estandarización para escalar los datos x. Como resultado, los datos estandarizados tendrán una distribución aproximadamente normal con una media de cero y una desviación estándar de uno.
Adicionalmente, usaremos una operación de **barajado** o mezcla aleatoria de los datos **x** y las etiquetas **y**, garantizando así que las instancias de datos y sus correspondientes etiquetas estén mezcladas aleatoriamente. Este proceso es útil para evitar sesgos o patrones no deseados que puedan surgir debido a un orden específico

6.1.2. Partición de los datos

En esta etapa del estudio, hay que aplicar metodologías de validación para la partición de los datos. Así, las metodologías de validación podemos utilizar la **validación cruzada** con 10-fold (**CV**) y la división de datos de entrenamiento-prueba (**Train-Test Split** con 70% para train).

La validación cruzada (CV) es una técnica que involucra la partición del conjunto de datos en 10 partes iguales, llamadas folds. En cada iteración, se selecciona un fold como conjunto de prueba y los 9 folds restantes se utilizan como conjunto de entrenamiento. Esto se repite 10 veces, de manera que cada fold se haya utilizado una vez como conjunto de prueba.

Por otro lado, la técnica de división de datos de entrenamiento-prueba (Train-Test Split) implica la separación del conjunto de datos en dos partes: un conjunto de entrenamiento y un conjunto de prueba. En este caso, utilizaremos una proporción del 70% para el conjunto de entrenamiento y el 30% restante se reservó para evaluar el rendimiento final del modelo.

6.1.3. Selección de modelos

Los algoritmos supervisados son técnicas de aprendizaje automático que se utilizan para entrenar modelos predictivos utilizando datos etiquetados, es decir, datos de entrada junto con las salidas o etiquetas correspondientes.

6.1.4. Random Forest

Random Forest funciona creando múltiples árboles de decisión en los que cada árbol es un clasificador que toma una decisión basada en un subconjunto aleatorio de características de los datos de entrenamiento.

En la ***Figura 11*** se observan los hiperparámetros establecidos para **Random Forest** basado en una de las mejores configuraciones. Mas adelante se hablará sobre el tema.

Mejores Hiperparámetros para Random Forest	
n_estimators	50
max_depth	50
min_samples_split	5
min_samples_leaf	2
criterion	gini

Figura 11 - Hiperparámetros utilizados.

En particular, utilizando **Cross Validation = 10**, en la ***Figura 12*** se muestran los resultados de las métricas evaluadas en porcentajes.

Random Forest						
CV	***Accuracy***	***Precisión***	***Recall***	***F1***	***AUC***	***Error***
1	83	81	86	83	83	17
2	82	82	83	82	82	18
3	79	80	79	79	79	21
4	85	84	86	85	85	15
5	83	81	87	84	83	17
6	80	78	85	81	80	20
7	82	82	82	82	82	18
8	81	79	85	82	81	19
9	84	81	87	84	83	17
10	82	81	84	83	82	18
Promedio	82	81	84	83	82	18
Tiempo de Ejecucion: 110. 93 seg						

Figura 12 - Resultados Random Forest Con CV=10.

Podemos inferir que el **Accuracy** ronda entre el *79 y 85*%, lo que indica que el modelo está clasificando correctamente en dichos porcentajes los exoplanetas en el conjunto de datos. La **Precisión** varía entre del *78 y 84*%; esto significa que, de todos los planetas clasificados como "*CONFIRMADO*", dichos porcentajes son los que realmente pertenecen a la clase "*CONFIRMADO*". El **Recall** ronda entre *79 y 87*%; esto indica que el modelo está identificando correctamente en dichos porcentajes los planetas que realmente son de la clase "CONFIRMADO" respecto al total. de casos positivos reales. El valor de **F1** ronda entre el *79 y 84*%; lo cual indica la medida de combinación de la **precisión** y el **recall**, proporcionando una medida balanceada de la calidad del modelo. Finalmente, el valor del **AUC** ronda entre el *79 y 85*%, representando el área bajo la curva de Característica Operativa del Receptor (**ROC**), y está midiendo el rendimiento del modelo en términos de su capacidad para distinguir entre las clases *CONFIRMADO* y *CANDIDATO*.

La **Figura 13** muestra el mejor **Accuracy** correspondiente al **CV=4**, para la matriz de confusión, donde la diagonal representa los *True Positive* para cada clase; en otras palabras, **True** como *CONFIRMADO* y **False** para *CANDIDATE*.

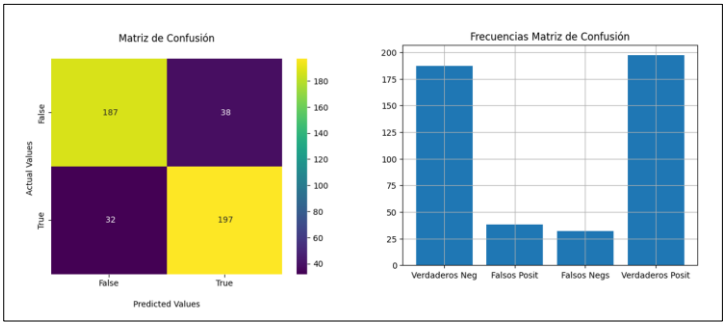


Figura 13 - Matriz de confusión.

Por otro lado, la **Figura 14**, muestra la comparación de valores **predichos** vs valores **teóricos**, siendo correctas las muestras en verde.

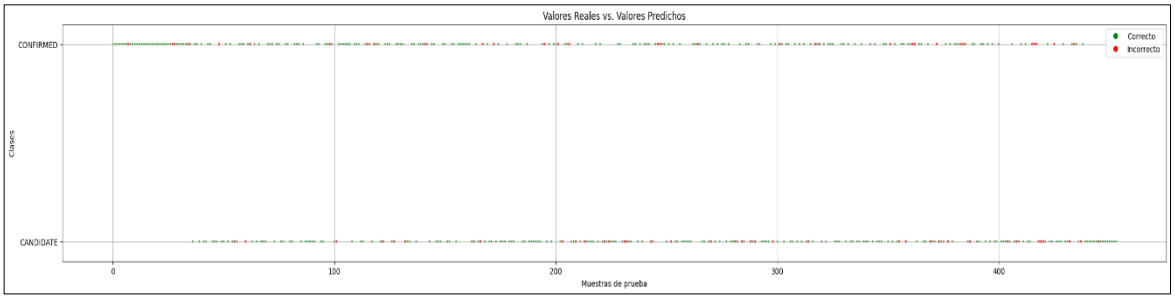


Figura 14 - Grafica Predichos Vs Teóricos.

Y en general, en la **Figura 15** se observan las curvas de aprendizaje tanto para los datos de **train** como **test**, y para cada métrica en cada iteración del CV.

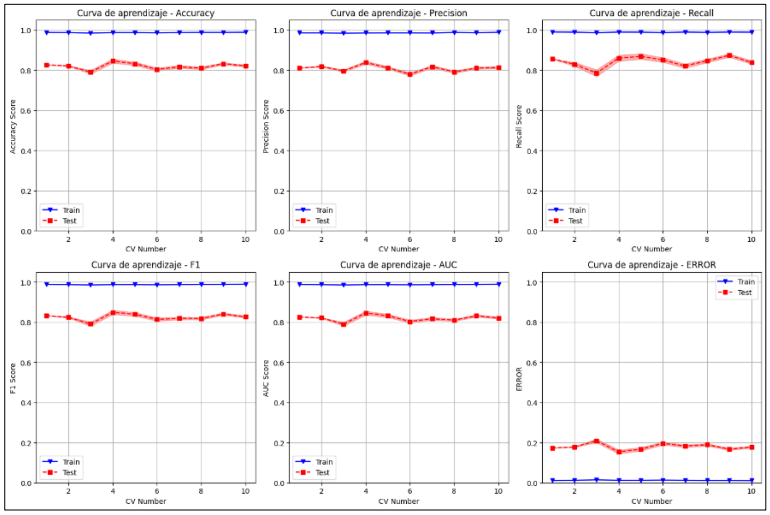


Figura 15 - Curvas de Aprendizaje.

Se puede observar que si bien, la curva de aprendizaje en este caso está basada en las distintas iteraciones del **CV,** inicialmente pareciera presenta una mejora en el desempeño del **test**, luego se presenta un comportamiento de *Bias*, puesto que en las distintas iteraciones parece ser que el error se estabiliza, mientras que en el **train** las distintas iteraciones no tienen variaciones.

Por el contrario, utilizando **Train-Test-Split = 70% para Train**, en la ***Figura 16*** se muestran los resultados de las métricas evaluadas en porcentajes. Además, se utilizaron iteraciones basadas en selección lotes para ciertos porcentajes del total de datos destinados para **train**.

Random Forest						
%	*Accuracy*	*Precisión*	*Recall*	*F1*	*AUC*	*Error*
10	80	76	84	80	80	20
20	79	76	82	79	79	21
30	79	76	83	79	79	21
40	81	78	84	81	81	19
50	81	79	84	81	81	19
60	81	78	84	81	81	19
70	81	79	84	81	81	19
80	81	78	85	81	81	19
90	81	78	85	82	81	19
100	81	78	83	81	81	19
Promedio	80	78	84	81	81	20
Tiempo de Ejecucion: 54. 33 seg						

Figura 16 - Resultados Random Forest con TTS=70% para Train.

La **Figura 17** muestra el mejor **Accuracy** correspondiente al **100%** de los datos destinados para **train** en la matriz de confusión.

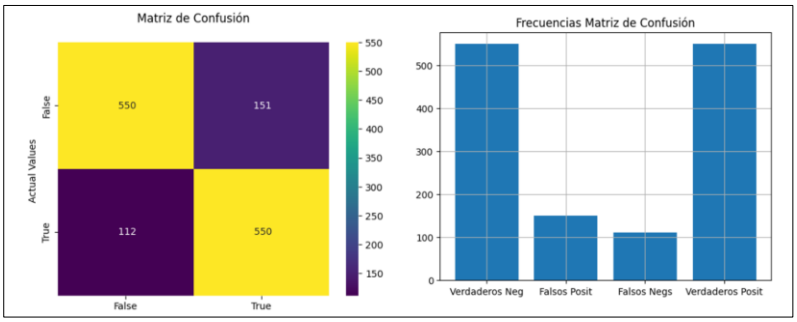


Figura 17 - Matriz de confusión.

Por otro lado, la **Figura 18**, muestra la comparación de valores **predichos** vs valores **teóricos**.

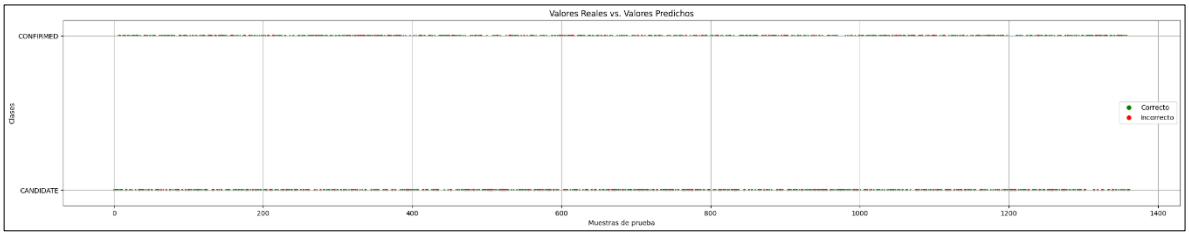


Figura 18 - Grafica Predichos Vs Teóricos.

Y en general, en la **Figura 19** se observan las curvas de aprendizaje tanto para los datos de **train** como **test**, y para cada métrica en cada selección de lote del **TTS**.

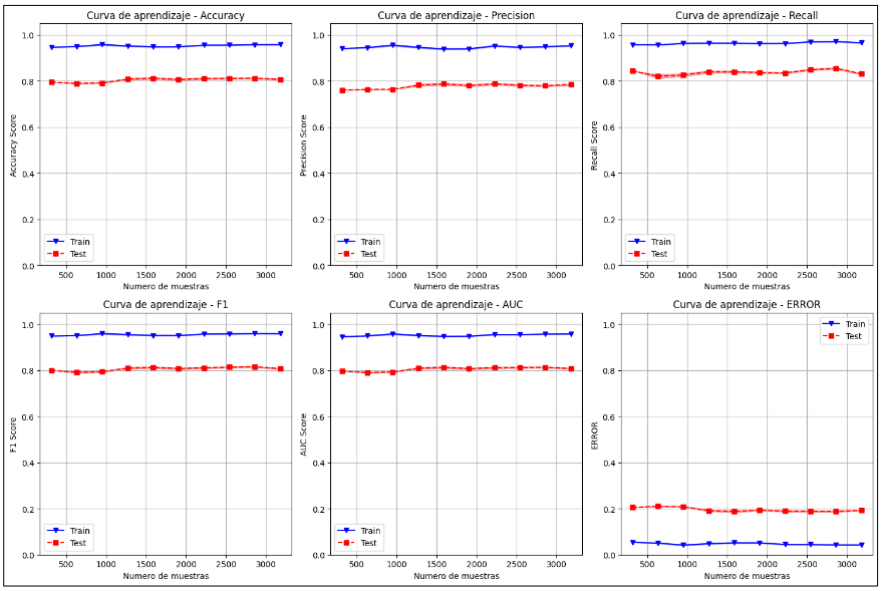


Figura 19 - Curvas de Aprendizaje

Se puede observar que si bien, la curva de aprendizaje inicialmente pareciera presentar una mejora en el desempeño del **test**, luego se presenta un comportamiento de *Bias*, puesto que a pesar de que se le dan más datos parece ser que el error se estabiliza y no disminuye a pesar de incrementar la cantidad de datos. Sin embargo, en el **train** la distinta selección de lotes no tiene variaciones.

6.1.5. Redes Neuronales

Las **redes neuronales** se construyen a partir de capas de neuronas artificiales que se comunican entre sí a través de conexiones. Cada neurona procesa la información de entrada utilizando una función de activación y transmite la salida a las neuronas de la capa siguiente.

Así, en una red neuronal, la información fluye hacia adelante a través de las capas, desde la capa de entrada hasta la capa de salida. Cada neurona en una capa está conectada a todas las neuronas de la capa siguiente mediante conexiones ponderadas, donde los pesos asociados a estas conexiones representan la importancia relativa de las entradas para la neurona.

En la ***Figura 19*** se observan los hiperparámetros establecidos para **Redes Neuronales** basado en una de las mejores configuraciones.

Mejores Hiperparámetros para Redes Neuronales	
optimizer	adam
hidden_units	[20, 10]
epochs	10
batch_size	5

Figura 19 - Hiperparámetros utilizados RN.

En particular, utilizando **Cross Validation = 10**, en la ***Figura 20*** se muestran los resultados de las métricas evaluadas en porcentajes.

Redes Neuronales						
CV	***Accuracy***	***Precisión***	***Recall***	***F1***	***AUC***	***Error***
1	83	81	86	84	83	17
2	84	84	84	84	84	16
3	80	82	77	79	80	20
4	85	84	88	86	85	15
5	83	81	87	84	83	17
6	87	85	89	87	87	13
7	87	87	87	87	87	13
8	85	84	87	86	85	15
9	88	87	89	88	88	12
10	87	84	90	87	87	13
Promedio	85	84	87	85	85	15
Tiempo de Ejecucion: 209. 34 seg						

Figura 20 - Resultados Redes Neuronales Con CV=10

La **Figura 21** muestra el mejor **Accuracy** correspondiente al **CV=9**, para la matriz de confusión, donde la diagonal representa los *True Positive* para cada clase.

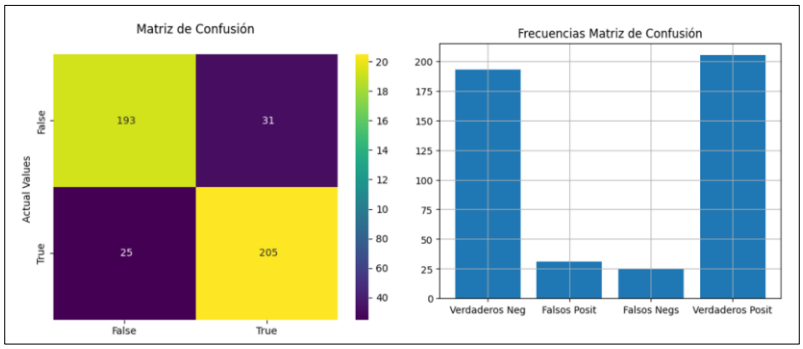


Figura 21 - Matriz de confusión

Por otro lado, la **Figura 14**, muestra la comparación de valores **predichos** vs valores **teóricos**, siendo correctas las muestras en verde.

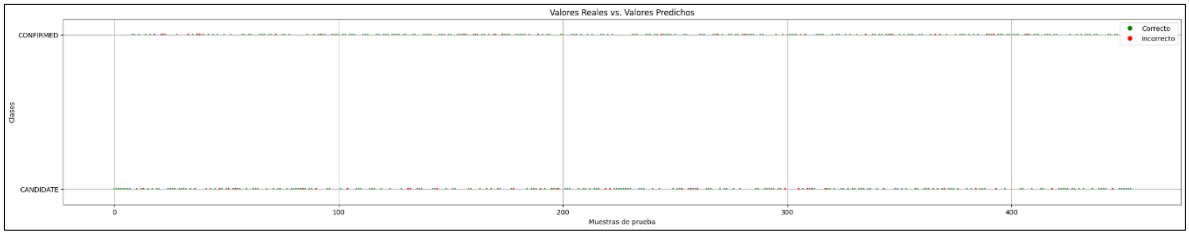


Figura 22 - Grafica Predichos Vs Teóricos

Y en general, en la **Figura 23** se observan las curvas de aprendizaje tanto para los datos de **train** como **test**, y para cada métrica en cada iteración del CV

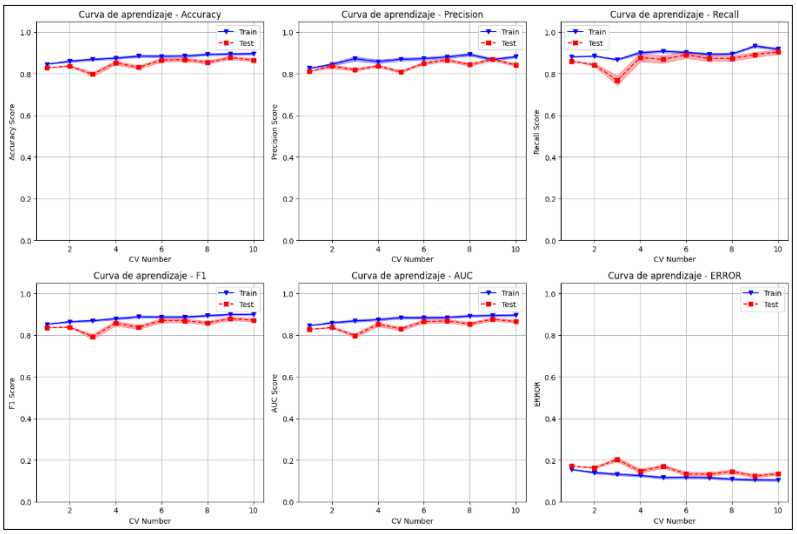


Figura 23 - Curvas de Aprendizaje

Por el contrario, utilizando **Train-Test-Split = 70% para Train**, en la ***Figura 24*** se muestran los resultados de las métricas evaluadas en porcentajes. Además, se utilizaron iteraciones basadas en selección lotes para ciertos porcentajes del total de datos destinados para **train**.

Redes Neuronales						
%	***Accuracy***	***Precisión***	***Recall***	***F1***	***AUC***	***Error***
10	77	74	81	77	77	23
20	79	76	82	79	79	21
30	80	75	88	81	80	20
40	81	79	83	87	81	19
50	81	80	82	81	81	19
60	80	81	78	79	80	20
70	80	79	82	80	81	20
80	80	79	80	79	80	20
90	81	80	83	81	81	19
100	81	78	85	81	81	19
Promedio	80	78	82	80	81	20
Tiempo de Ejecucion: 125. 92 seg						

Figura 24 - Resultados Redes Neuronales con TTS=70% para Train.

La **Figura 25** muestra el mejor **Accuracy** correspondiente al **100%** de los datos destinados para **train** en la matriz de confusión.

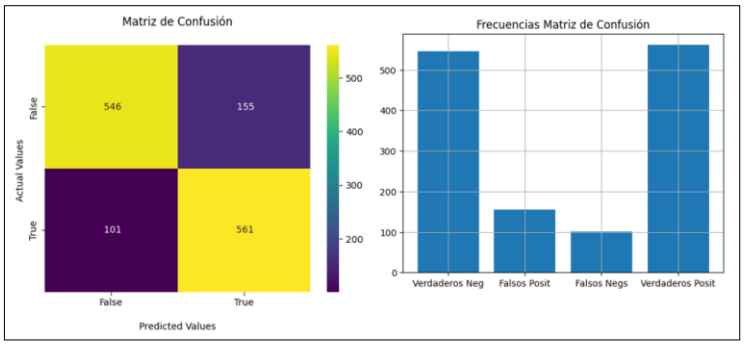


Figura 25 - Matriz de confusión.

Por otro lado, la **Figura 26**, muestra la comparación de valores **predichos** vs valores **teóricos**.

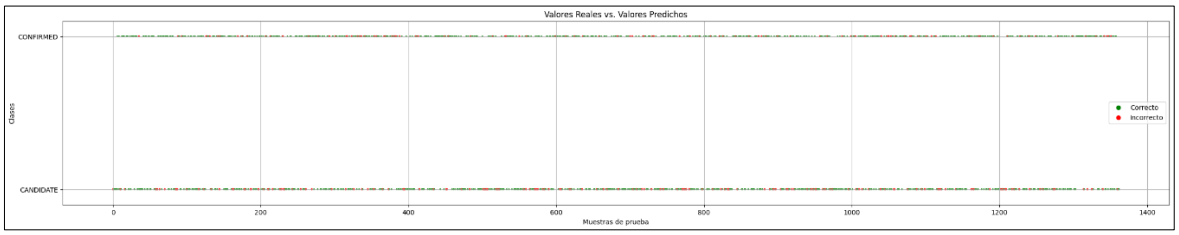


Figura 26 - Grafica Predichos Vs Teóricos.

Y en general, en la **Figura 27** se observan las curvas de aprendizaje tanto para los datos de **train** como **test**, y para cada métrica en cada selección de lote del **TTS**.

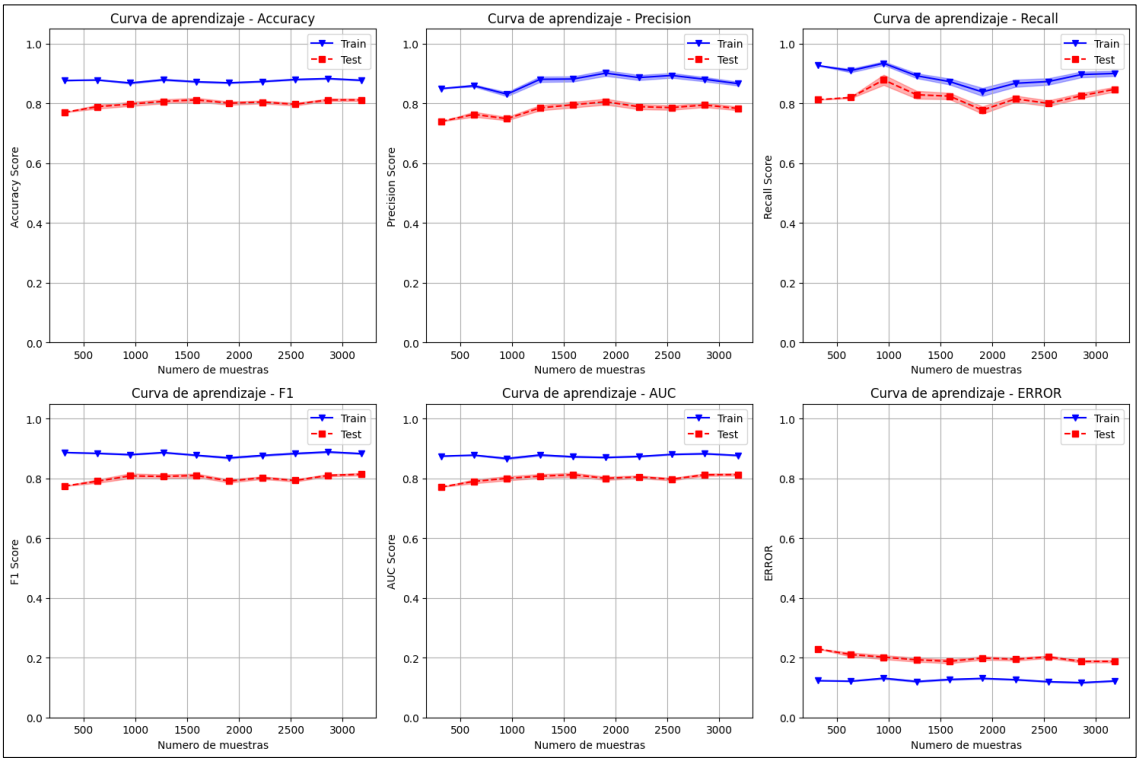


Figura 27 - Curvas de Aprendizaje

En restrospectiva, el rendimiento de los modelos **RF** y **RN** utilizando técnicas de evaluación como Cross Validation y Train-Test Split ha sido sólido, con puntajes consistentes en el rango del 75% al 85%. Esto indica que los modelos son capaces de clasificar correctamente entre las clases objetivo para clasificar exoplanetas como, ***CANDIDATO*** y ***CONFIRMADO***. Además, la variabilidad observada en los puntajes de rendimiento puede estar influenciada por varios factores, como la misma complejidad del conjunto de datos, la selección de características, los hiperparámetros y la aleatoriedad inherente en los algoritmos de **RF** y **RN.**

6.2. Mejores hiperparámetros de los modelos

Para intentar obtener una adecuada selección de hiperparámetros para los modelos a trabajar, se hace un testeo exhaustivo mediante el módulo de la librería de *sk.learn.model_selection*, llamado ***GridSearchCV***. Este módulo permite hacer un análisis variando los hiperparámetros del algoritmo de interés implementando una metodología de *cross-validation*. Como resultado se obtienen los mejores hiperparámetros entre una selección de parámetros dados; además utilizando un enfoque para parámetros no muy complejos para los tiempos de ejecución.

Por su parte, en la ***Figura 28*** se muestran los hiperparámetros usados para **RandomForest**.

Lista de Hiperparámetros para Random Forest				
n_estimators	5	20	30	50
max_depth	5	10	20	50
min_samples_split	2	5	10	20
min_samples_leaf	2	5	10	20
criterion	entropy	gini		

Figura 28 - Lista de Hiperparámetros para RF.

Finalmente, en la ***Figura 29*** se obtienen los mejores hiperparámetros para **Random Forest**

Mejores Hiperparámetros para Random Forest	
n_estimators	50
max_depth	50
min_samples_split	5
min_samples_leaf	2
criterion	gini
Métricas correspondientes	
Accuracy	82
Precisión	81
Recall	88
F1	83
AUC	90

Figura 29 - Mejores Hiperparámetros para RF.

A su vez, en la ***Figura 30,*** se muestran las **curvas de validación** con cada uno de los hiperparámetros probados; evidenciando que si se aumenta el numero de unidades, mejora considerablemente el rendimiento del modelo, ya sea para ***n_estimators*** o ***max_depth***.
Sin embargo, para hiperparámetros como ***min_samples_split*** o ***min_samples_leaf***, se evidencia que aumentando el número de unidades, el desempeño del modelo cae tanto para **train**, como **test,** pero siendo más significativa para los datos de test.

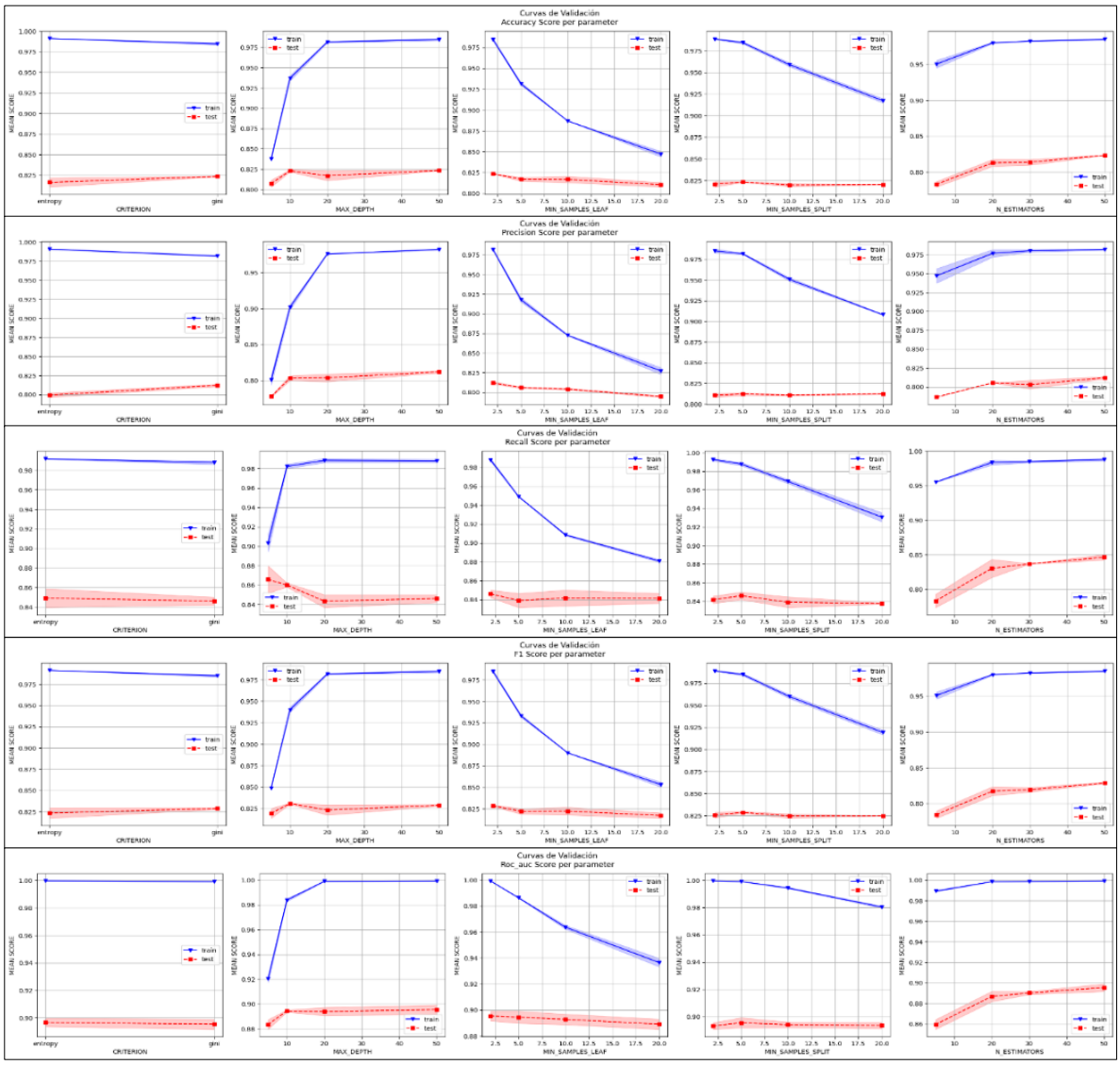


Figura 30 - Curvas de Validación RF.

Por el contrario, para Redes Neuronales se establecieron los siguientes parámetros en la ***Figura 30.***

Lista de Hiperparámetros para Redes Neuronales				
optimizer	adam	rmsprop		
hidden_units	[20, 10]	[10, 5]	[30, 5]	
epochs	10	50	100	
batch_size	5	15	32	100

Figura 31 - Lista de Hiperparámetros para RN.

Finalmente, en la ***Figura 32*** se obtienen los mejores hiperparámetros para **Redes Neuronales.**

Mejores Hiperparámetros para Redes Neuronales	
optimizer	adam
hidden_units	[20, 10]
epochs	10
batch_size	5
Métricas correspondientes	
Accuracy	85
Precisión	82
Recall	81
F1	87
AUC	90

Figura 32 - Mejores Hiperparámetros para RN.

De esta manera, en la ***Figura 33,*** se muestran las **curvas de validación** con cada uno de los hiperparámetros probados; evidenciando que si se aumenta el número de **lotes**, para entrenar la red, un alto número deja de ser relevante; o que en todos los casos el optimizado **Adam** es el mejor. Así como que hay un mal desempeño cuando se utilizan pocas neuronas en las capas ocultas.

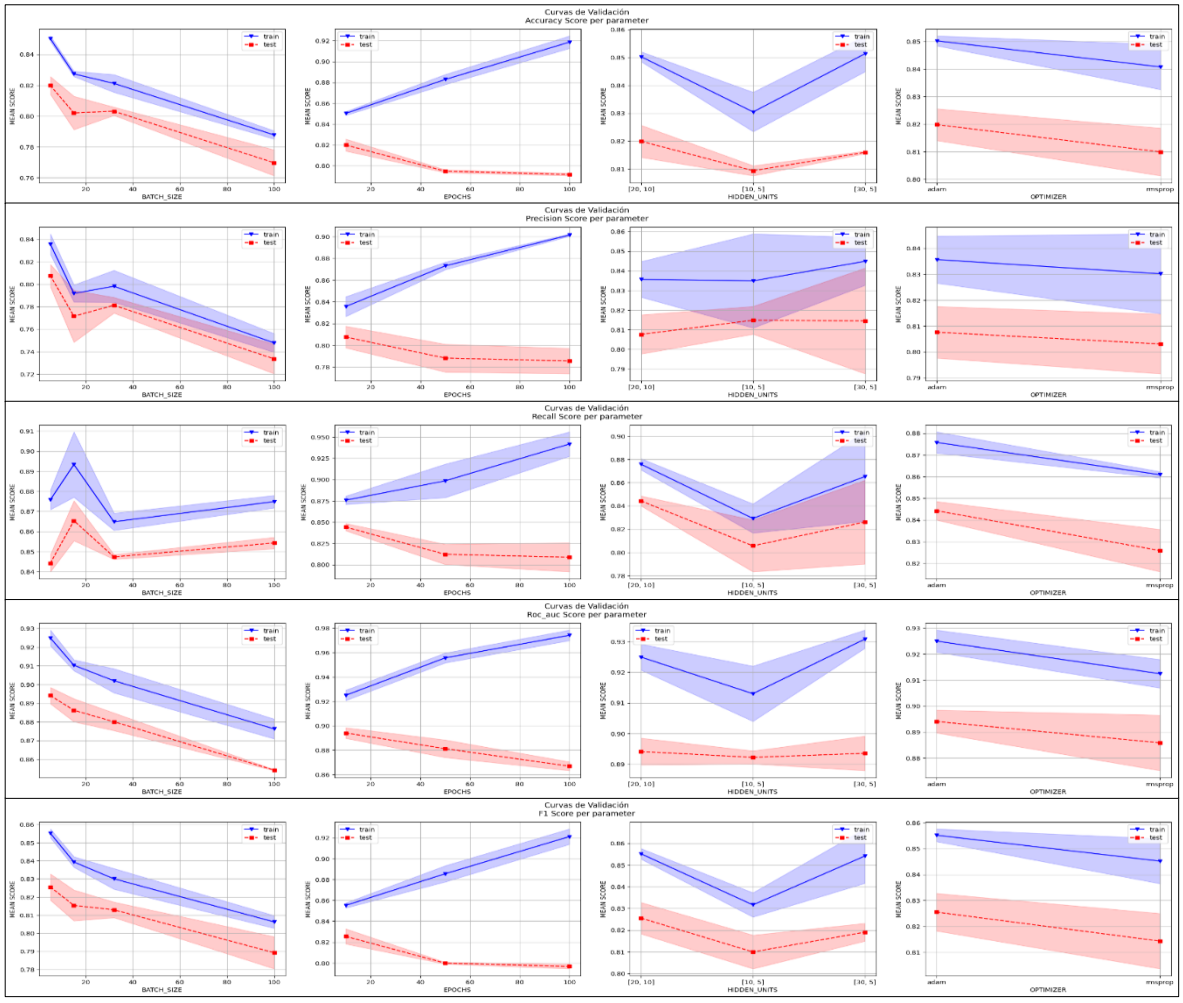


Figura 33 - Curvas de Validación RN.

7. Algoritmos no supervisados

7.1.1. Selección de modelos

Los métodos no supervisados son técnicas de aprendizaje automático que se utilizan para explorar y analizar datos sin la presencia de etiquetas o información de salida previamente conocida. A diferencia de los métodos supervisados, en los que se tienen datos de entrada y salidas correspondientes para entrenar un modelo, los métodos no supervisados se centran en encontrar patrones, estructuras o relaciones intrínsecas en los datos sin la necesidad de una guía externa.

7.1.2. K-Means

K-Means: Utilizado para clustering o agrupamiento de datos. Funciona dividiendo los datos en k grupos o clusters, donde k se define previamente. El algoritmo asigna cada punto de datos al cluster más cercano y luego recalcula el centro de cada cluster.

En particular, se podría utilizar una agrupación en 2 clusters que correspondería a nuestro problema de clasificación biclase, e intentar saber si por entrenamiento no supervisado y por la naturaleza de los datos, los exoplanetas podrían diferenciarse de *CANDIDATE* Y *CONFIRMED*.
Sin embargo, en la **Figura 34** se muestran los resultados del agrupamiento.

K-Means=2 Clusters	
Silhouette score	0.93
Davies-Bouldin score:	0.56
Calinski-Harabasz score	429
Tiempo de Ejecucion	325 seg

Figura 34 - Resultados K-Means.

Donde:

Silhouette score es el indice que mide cuán similares son los objetos dentro de un grupo y cuán diferentes son de los objetos de otros grupos. El puntaje varía de -1 a 1, donde valores más cercanos a 1 indican que los grupos están bien separados y los objetos dentro de cada grupo están muy juntos.

Davies-Bouldin score es otro índice que mide la calidad del clustering, basado en la distancia entre los centroides de los grupos y la distancia entre los objetos dentro de cada grupo. Un puntaje más bajo indica una mejor separación de los grupos.

Calinski-Harabasz score: Este es un índice que mide la relación entre la varianza dentro de cada grupo y la varianza entre los grupos. Cuanto mayor sea el puntaje, mejor será la calidad del clustering.

En consecuencia, como es imposible graficar la distribución de los clusters, porque tenemos 44 características, y porque el tiempo de ejecución es elevado podemos optar por utilizar técnicas de **entrenamiento no supervisado** basadas en el tratamiento de los datos antes del entrenamiento.

7.1.3. PCA + K-Means

PCA es utilizado para reducir la dimensionalidad de un conjunto de datos. Funciona proyectando los datos en un espacio de menor dimensión que aún mantiene la mayor cantidad posible de variabilidad de los datos originales.

Ahora, para encontrar el número óptimo de componentes se determina considerando la varianza explicada acumulada. La varianza explicada acumulada muestra la cantidad total de varianza en los datos que se puede explicar utilizando un número creciente de componentes principales. La elección del número óptimo de componentes depende de la cantidad de varianza que se desea retener en los datos.

Para nuestro caso, debido al alto número de características presentes en el dataset, podríamos aplicar PCA para reducir la dimensionalidad e inventar visualizar en 2 dimensiones el agrupamiento de los datos al usar K-Means. Sin embargo, tener presente que el PCA transforma las características en valores no interpretables y que además reducir la dimensión puede generar pérdida de información relevante para el dataset; por ello, optamos por encontrar la varianza explicada acumulada.

Así, en la **Figura 35** se muestran los resultados utilizando sólo 2 componentes.

PCA + K-*Means=2 Componentes y 2 Clusters*	
Silhouette score	0.99
Davies-Bouldin score:	0.0002
Calinski-Harabasz score	138571.8
Tiempo de Ejecucion	1.6 seg

Figura 35 - Resultados PCA + K-Means.

En la ***Figura 36***, podemos observar la distribución del agrupamiento. En primera instancia podríamos pensar que las métricas son adecuadas (**Figura 35**), pero es extraño que sólo agrupe 1 muestra en 1 sólo cluster (***Figura 36)***, podría ser problemas de la selección de componentes.

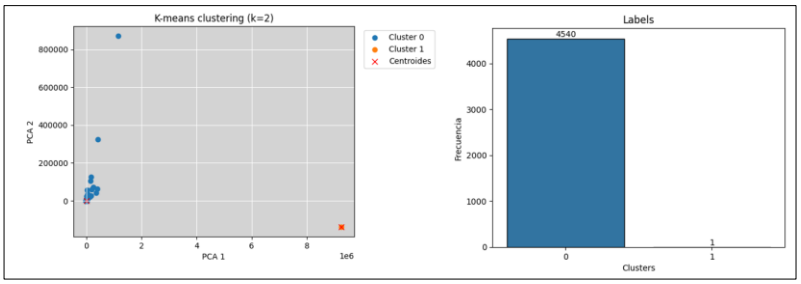


Figura 36 - Agrupamiento PCA + K-Means.

Si calculamos la ***varianza explicada acumulada*** en un 80%, obtenemos que el número óptimo de componentes es **18**. (**Figura 37**).

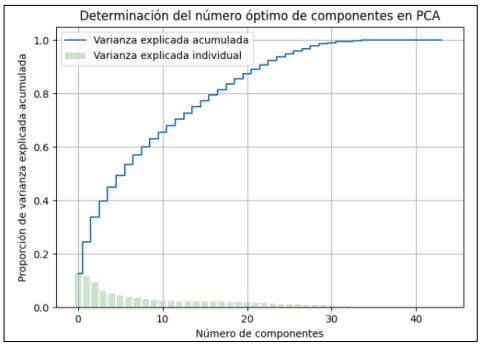


Figura 37 - Numero óptimo de componentes PCA.

De esta manera, en las **Figura 38** se muestran los resultados utilizando los 18 componentes y agrupando en 18 clusters. La **Figura 39** en el gráfico de barras representa mejor la Frecuencia de clusters; reiterando que tenemos 18 componentes, pero sólo graficamos 2.

PCA + K-*Means=18 Componentes y 18 Clusters*	
Silhouette score	0.18
Davies-Bouldin score:	0.85
Calinski-Harabasz score	612.42
Tiempo de Ejecucion	2.9 seg

Figura 38 - Resultados PCA + K-Means

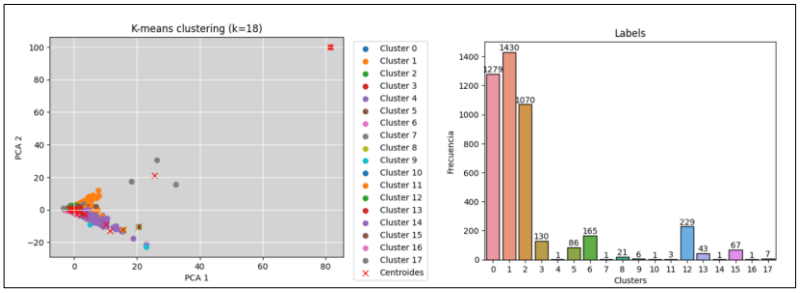


Figura 39 - Agrupamiento PCA + K-Means.

Sin embargo, no es evidencia clara de la clasificación biclase que queremos.

7.1.4. T-SNE + K-Means

El **t-SNE** (t-Distributed Stochastic Neighbor Embedding) es un algoritmo de reducción de dimensionalidad no lineal utilizado para visualizar y explorar datos de alta dimensionalidad. A diferencia del PCA, que es un método lineal, el t-SNE es capaz de capturar relaciones no lineales entre las muestras y preservar estructuras complejas en los datos. Además para t-SNE no existe una rúbrica específica o una regla general para determinar el número óptimo de componentes , todo dependerá de la naturaleza del dataset.
Ahora, en las **Figura 40** y **41** se muestran los resultados utilizando también 2 componentes y agrupando en 2 clusters.

tSNE + K-Means=2 Componentes y 2 Clusters	
Silhouette score	0.35
Davies-Bouldin score:	1.19
Calinski-Harabasz score	50.18
Tiempo de Ejecucion	50.18 seg

Figura 40 - Resultados tSNE + K-Means

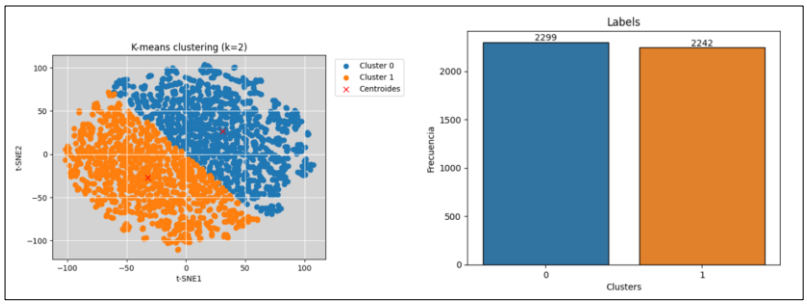


Figura 41 - Agrupamiento tSNE + K-Means.

Pareciera que tSNE obedece a la clasificación biclase que queremos buscar, además al encontrar una candidad de datos equilibrada en cada cluster. Sin embargo, se puede notar un evidente solapamiento de datos, donde posiblemente podrían estar contenidos en 1 sólo cluster.

8. Algoritmos no supervisados y supervisados

8.1.1. Selección de modelos

Una vez realizado un análisis solamente con **algoritmos supervisados** y desarrollar un análisis implementando **algoritmos no supervisados** a modo de tratamiento de los datos y agrupamiento en clusters; ahora podemos probar la combinación de ambos. De esta manera reduciremos la dimensionalidad del dataset con **PCA** y **tSNE** y haremos aprendizaje supervisado, con **RF** y **RN.**

8.1.2. PCA + Random Forest

De acuerdo a **Random Forest**, para ajustar sus hiperparámetros ya utilizando un número de componentes para **PCA=18** y utilizando **Cross Validation = 10**, de nuevo se realizó una búsqueda de hiperparámetros, obteniendo que los mejores son: ***{'criterion': 'entropy', 'max_depth': 20, 'min_samples_leaf': 2, 'min_samples_split': 5, 'n_estimators': 50}***

La ***Figura 42*** se muestran los resultados de las métricas evaluadas.

PCA + Random Forest						
CV	Accuracy	Precisión	Recall	F1	AUC	Error
1	78	76	81	79	78	22
2	78	79	79	79	78	22
3	74	74	76	75	74	26
4	78	78	78	78	78	22
5	77	75	82	78	77	23
6	77	76	81	78	77	23
7	77	76	79	78	77	23
8	77	76	79	78	77	23
9	79	78	81	80	79	21
10	76	75	77	76	76	24
Promedio	77	76	79	78	77	23
Tiempo de Ejecucion: 80.40 seg						

Figura 42 - Resultados PCA + Random Forest y CV.

Por el contrario, utilizando de nuevo **Train-Test-Split = 70% para Train**, en la ***Figura 43*** se muestran los resultados de las métricas evaluadas. Conservando también la configuración anterior.

PCA + Random Forest						
%	Accuracy	Precisión	Recall	F1	AUC	Error
10	79	75	85	80	79	21
20	79	76	83	79	79	21
30	80	76	85	80	80	20
40	81	80	82	81	81	19
50	81	79	84	81	81	19
60	81	79	83	81	81	19
70	81	78	84	81	81	19
80	80	77	85	80	81	20
90	81	78	85	81	81	19
100	81	78	85	82	81	19
Promedio	80	78	84	81	81	20
Tiempo de Ejecucion: 42. 98 seg						

Figura 43 - Resultados PCA + Random Forest y TTS.

8.1.3. T-SNE + Redes Neuronales

De acuerdo a **Redes Neuronales,** para ajustar sus hiperparámetros ya utilizando un número de componentes para **t-SNE=2** y utilizando **Cross Validation = 10**, de nuevo se realizó una búsqueda de hiperparámetros, obteniendo que los mejores son: ***{'batch_size': 5, 'epochs': 100, 'hidden_units': [20, 10], 'optimizer': 'adam'}***

La ***Figura 44*** se muestran los resultados de las métricas evaluadas.

tSE + Redes Neuronales						
CV	**Accuracy**	**Precisión**	**Recall**	**F1**	**AUC**	**Error**
1	64	60	87	71	64	36
2	69	65	82	73	69	31
3	65	60	89	72	65	35
4	70	69	73	71	70	30
5	66	64	74	69	66	34
6	67	67	68	67	67	33
7	69	69	71	70	69	31
8	70	69	73	71	70	30
9	69	68	72	70	69	31
10	70	67	83	74	70	30
Promedio	68	66	77	71	68	32
Tiempo de Ejecucion: 344. 64 seg						

Figura 44 - Resultados tSNE + Redes Neuronales y CV.

Por el contrario, utilizando de nuevo **Train-Test-Split = 70% para Train**, en la ***Figura 45*** se muestran los resultados de las métricas evaluadas. Conservando también la configuración anterior.

tSNE + Redes Neuronales						
%	**Accuracy**	**Precisión**	**Recall**	**F1**	**AUC**	**Error**
10	60	58	64	61	60	40
20	64	61	70	65	64	36
30	61	56	92	70	62	39
40	64	61	71	66	64	36
50	63	58	82	68	63	37
60	64	61	69	65	64	36
70	66	65	63	64	65	34
80	64	59	84	69	65	36
90	66	63	70	66	66	34
100	65	61	80	69	66	35
Promedio	64	60	74	66	64	36
Tiempo de Ejecucion: 241. 42 seg						

Figura 45 - Resultados tSNE + Redes Neuronales y TTS.

En general, podemos decir que después de realizar una comparación entre el uso de **PCA** junto con **Random Forest** y **t-SNE** con **Redes Neuronales**, se han observado resultados más pobres en términos de métricas de rendimiento, en comparación con el uso individual de **RF** y **RN**. Aunque el tiempo de ejecución se ha reducido al utilizar técnicas de reducción de dimensionalidad, los resultados generales no son satisfactorios.

Las causas podrían ser por efecto de la reducción de dimensionalidad, que puede llevar a una pérdida de información importante necesaria para un rendimiento óptimo del modelo, o que los algoritmos de aprendizaje automático pueden tener diferentes niveles de sensibilidad ante la reducción de dimensionalidad, o que se necesita un mejor ajuste a los hiperparámetros, así como considerar la aplicación de otras técnicas de reducción de dimensionalidad, o también realizar evaluaciones exhaustivas y comparativas de diferentes combinaciones de técnicas de reducción de dimensionalidad y algoritmos de aprendizaje automático.

9. Retos y condiciones de despliegue del modelo

Como retos y condiciones, un posible despliegue del modelo debe garantizar:
Escalabilidad: Si el dataset de exoplanetas sigue creciendo con nuevas observaciones, es importante asegurarse de que el modelo pueda manejar eficientemente un mayor volumen de datos.

Eficiencia computacional: Al desplegar el modelo, es esencial que pueda ejecutarse rápidamente y de manera eficiente. Esto implica optimizar el código y utilizar técnicas de procesamiento paralelo si es necesario para reducir el tiempo de predicción y garantizar una respuesta rápida del modelo.

Actualización del modelo: A medida que se recopilen nuevos datos o se realicen mejoras en los algoritmos de clasificación, debería de ser necesario actualizar el modelo existente, por ello se requeriría actualización y reentrenamiento del modelo de manera regular para mantener su rendimiento.

Monitorización y mantenimiento: Una vez que el modelo estuviera en producción, debería de establecer mecanismos de monitorización para evaluar su rendimiento y detectar posibles problemas o degradaciones en su desempeño

Seguridad y privacidad: Si el dataset de exoplanetas contiene información confidencial o sensible, se debería de aplicar medidas de seguridad y protección de la privacidad.

En retrospectiva, si resulta factible utilizar modelos de machine learning para la detección o confirmación de exoplanetas; porque podremos mejorar las técnicas de identificación de patrones y características en los datos que no son evidentes para los científicos a simple vista, como las técnicas de detección de e identificación de nuevos exoplanetas que de otra manera podrían haber pasado desapercibidos.

10. Conclusiones

Con un vistazo general, podemos contemplar que para todo aprendizaje, si se desean tener unos excelentes resultados se requieren de fuertes habilidades y recursos, técnicas que hasta nuestros días se perfeccionan poco a poco, brindando un mundo de conocimiento que crece constantemente.
En este proceso en particular se obtuvieron resultados que no terminan de convencer y que deberían de ser mejores, en gran parte todo radica por la información contenida en el dataset, pudiendo ser muy compleja.

Pudimos comprobar con este proyecto que los modelos de aprendizaje con Machine Learning tienen su aplicación en un largo espacio, prácticamente en toda actividad investigativa estamos aplicándolo, porque como herramienta permite ahorrar tiempo, dinero y esfuerzo en algunos aspectos que de llevarlos luego a la práctica nos causarían muchos errores costosos y peligrosos.

Por otra parte, para la exploración espacial, sugiere que los modelos de aprendizaje puedan ser utilizados en potencia para detectar y clasificar cuerpos celestes quizá desconocidos e incomprensibles hasta la fecha. A partir de ahí, se podrán identificar posibles candidatos para misiones futuras, dirigir los recursos de manera más efectiva y en última instancia, mejorar nuestras capacidades para explorar el universo.

11. Bibliografía

[1] Kepler Objects of Interest. https://exoplanetarchive.ipac.caltech.edu/cgi-bin/TblView/nph-tblView?app=ExoTbls&config=koi

[2] Nasa. “Kepler Exoplanet Search Results.” Kaggle, 10 Oct. 2017. www.kaggle.com/nasa/kepler-exoplanet-search-results.

[3] “Overview.” NASA, NASA, 2 Apr. 2021. https://exoplanets.nasa.gov/what-is-an-exoplanet/planet-types/overview/

[4] Malik, A. P. Moster, B. Obermeier C “Exoplanet Detection using Machine Learning.” https://arxiv.org/pdf/2011.14135.pdf

[5] Logan “Identifying Exoplanets using Multiple Classification Models.” Jul 31, 2021. https://medium.com/@scheidlogan/identifying-exoplanets-using-multiple-classification-models-7ee48024d7fd

[6] Clayton, G. Manry, B. Rafiqi, S “Machine Learning Pipeline for Exoplanet Classification” 2019 https://scholar.smu.edu/cgi/viewcontent.cgi?article=1070&context=datasciencereview

[7] mdredyanahmed “Visualization-of-Kepler-Exoplanet-Search-Results.” Apr 12, 2021. https://github.com/mdredyanahmed/Visualization-of-Kepler-Exoplanet-Search-Results

[8] NASA “Exoplanet Watch Results.” https://exoplanets.nasa.gov/exoplanet-watch/results/