Universidad de Chile
Facultad de Ciencias Físicas y Matemáticas
Departamento de Ingeniería Eléctrica

Entrega Final

Clasificación de supernovas de ALeRCE - ZTF

Integrantes: Benjamín Irarrázabal T.
Joaquín Zepeda V.
Profesor: Pablo Estevez V.
Auxiliar: Ignacio Reyes J.
Ayudantes: Daniel Baeza M.
Roberto Cholaky M.
Francisca Cona F.
Bastián G. Labbé
Andrés González F.
Javier Molina F.
Pablo Montero S.
Óscar Pimentel
Tutor: Pablo Montero S.

Fecha de realización: 16 de diciembre de 2021
Fecha de entrega: 29 de octubre de 2021
Santiago de Chile

Índice de Contenidos

Índice de Figuras

Índice de Tablas

Índice de Códigos

1. Motivación y Descripción del Problema

La observación del cielo, el movimiento de los astros y de otros objetos astronómicos ha sido una labor muy importante del ser humano, realizando predicciones sobre sucesos que pudiesen ocurrir en un futuro ya sea próximo o lejano y dentro de esto, es donde se puede encontrar a las curvas de luz, que son series de tiempo astronómicas con información sobre la magnitud del brillo estelar, el error asociado y el día en que se presentó la alerta. No obstante, existen distintos tipos de curvas y según la primera versión de ALeRCE (*Automatic Learning for the Rapid Classification of Events*), estas se pueden clasificar en 15 subclases distintas, dentro de las cuales se encuentran distintos tipos de estrellas y otros objetos astronómicos, como por ejemplo, las supernovas. Estas últimas, son caracterizadas por ALeRCE como un objeto transiente y que según *Oxford Languajes* corresponden a una "estrella en explosión que libera una gran cantidad de energía; se manifiesta por un aumento notable de la intensidad del brillo o por su aparición en un punto del espacio vacío aparentemente".
Las supernovas también se pueden dividir en subclases dependiendo de su tipo y en esto será enfocado el siguiente proyecto, el cual tiene como finalidad clasificar supernovas mediante un modelo supervisado para lograr separarlas según su clase (SNIa, SNII, SNIIb, SNIIn, SLSN, SNIbc), sobre las que se profundizará más adelante.

2. Objetivos del Proyecto

Como fue mencionado en la sección anterior, el proyecto se enfocará en crear un modelo supervisado que logre clasificar las supernovas de acuerdo a su clase (tipo), además de esto se realizara la creación de un modelo supervisado que clasifique supernovas tipo SNIa y NoSNIa. Esto se realizará mediante la inspección de las muestras provenientes del survey astronómico Zwicky Transient Facility y luego usar el extractor de características provisto por el equipo docente para proponer el modelo. Además, se estudiarán dos algoritmos de clasificación, dentro de los cuales se escogerá el más óptimo para este caso.

3. Base de Datos

Como fue mencionado, la base de datos corresponde a las muestras provenientes del survey astronómico *ZTF*, estas contienen información sobre la magnitud del brillo, su error, el tiempo en el que ocurrió en una unidad de medida astronómica denominada *Modified Julian Date*, entre otras, de un total de 2068 supernovas distintas, datos que se utilizarán para extraer las características de las supernovas e implementar el clasificador. Las muestras que incluye la base de datos corresponde a alertas del tipo transiente [1], en específico de supernovas SNIa, SNII, SNIIb, SNIIn, SLSN, SNIbc, como será comprobado más adelante.
Esta base de datos poseía dentro de sus columnas "magpsf_corr" y "sigmapsf_corr" datos de tipo NaN, estos ocasionan problemas para realizar la extracción de características y por ende se deben corregir. Para esto, se tienen dos opciones, eliminar la fila que posea algún valor de este tipo o "manipular" la base de datos, reemplazando estos valores por sus respectivos de las columnas "magpsf" y "sigmapsf" (correspondientes a la magnitud obtenida por fotometría y su error asociado respectivamente). En base a esto, el equipo opta por la segunda opción para evitar la pérdida de aproximadamente 77000 muestras, lo cual hubiese sido perjudicial para el entrenamiento y la

clasificación.

4. Procesamiento de Datos

Como fue mencionado en la sección 3, el equipo para poder realizar una correcta extracción de características debió realizar una previa "limpieza de datos", la cual consistió en corregir los valores de tipo NaN para posteriormente elegir las columnas más importantes dentro de este conjunto y finalmente eliminar posibles outliers en las muestras. Para la detección de outliers se utilizo el indicador estadístico **z-score**, el cual permite comparar los datos y obtener el número de desviaciones típicas que el registro toma con respecto a la media, de esta manera los registros que tenían un valor de z-score mayor a 3, fueron descartados e identificados como outliers, esto se interpreta que los valores que están a más de 3 desviaciones estándar del promedio fueron descartados. Luego de este procedimiento se eliminan 1071 outliers y se conservan 127647 muestras de 2068 supernovas distintas. Estas, se encuentran desbalanceadas, siendo las de tipo Ia las más comunes, la comparación entre la cantidad de clases se puede observar a continuación.

Tabla 1: Tabla comparativa de las clases de supernovas presentes en la base de datos

Clase	Cantidad	Porcentaje c/r al total
SLSN	34	1.64 %
SNII	330	15.96 %
SNIIb	16	0.77 %
SNIIn	42	2.03 %
SNIa	1540	74.47 %
SNIbc	106	5.13 %

Y si observamos estos datos de manera gráfica se podrá notar el gran desbalance presente en el conjunto.

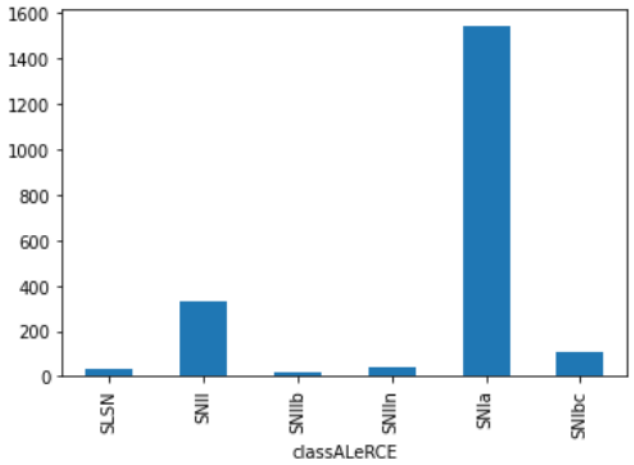


Figura 1: Histograma con las clases presentes en la base de datos

En la figura 1, se puede notar la clara predominancia de la clase SNIa con respecto a las otras, alcanzando un 74.47 % del total.

Las columnas que se utilizan en los extractores son las siguientes:

- magpsf: magnitud obtenida a partir de la fotometría que uso flujo de diferencia
- magpsf_corr: magnitud obtenida a partir de fotometría que suma flujo de diferencia con flujo de referencia.
- mjd : Modified Julian Date (días).
- fid: Indica la banda a la que corresponde (1 corresponde a green,2 corresponde a red).
- sigmapsf: corresponde al error de magpsf.
- sigmapsf_corr: corresponde al error de magpsf_corr.

Luego de esto, se realizo el proceso de extracción de características de las curvas de luz, el proceso extracción de características toma al rededor de 20 a 25 minutos. Para esto se utilizaron los extractores provistos por la librería de lc_classifier provista por ALeRCE, la cual permite extraer mediante extractores diferentes características que permiten realizar posteriormente el proceso de clasificación. Los extractores utilizados hasta ahora corresponden a:

- SupernovaeDetectionFeatureExtractor
- PeriodExtractor
- MHPSExtractor
- GPDRWExtractor
- FoldedKimExtractor
- HarmonicsExtractor
- PowerRateExtractor
- TurboFatsFeatureExtractor
- SNParametricModelExtractor

Uno de los extractores más importantes corresponde al **SNParametricModelExtractor**, el cual ajusta un modelo paramétrico SNe a la curva de luz y proporciona los parámetros ajustados como características.

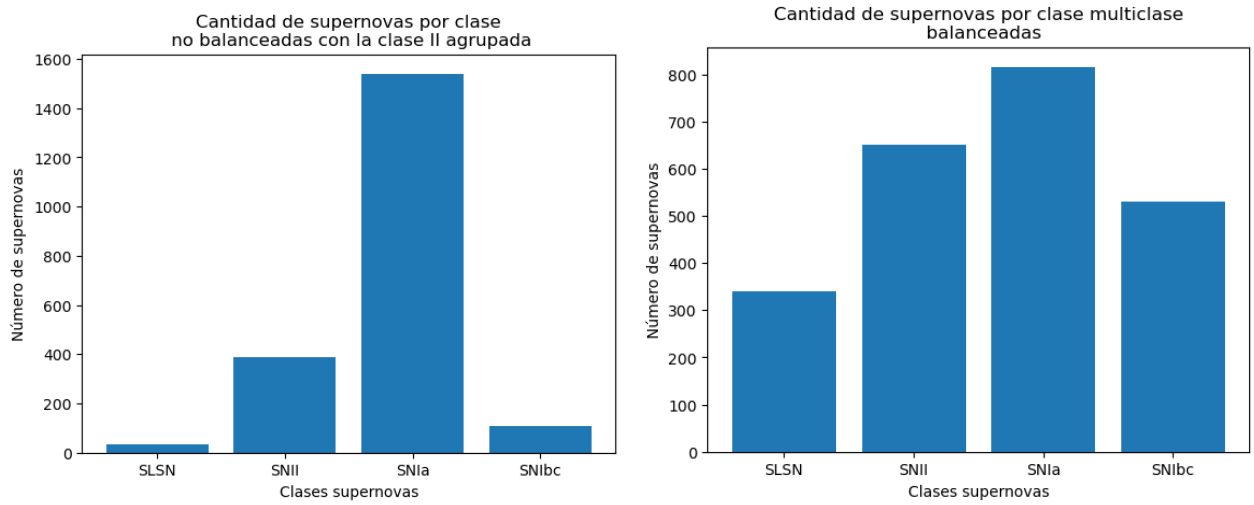
Luego de este proceso de extracción, se obtuvieron **142 características**, de las cuales se eliminan las características no recomendadas por el tutor, quedando un total de **92 características/features**.

Después de la extracción se arreglaron nuevamente los datos Nan, por lo que de los 2068 registros quedaron **1732** en total. Es importante notar que este valor depende del número de características que se consideran para la clasificación, y se elige un número menor de características este número aumenta, a continuación se realizara clasificación con una selección de 19 características las cuales cuentan con un total de **1873** registros (supernovas con su respectivas características y su respectivo target).

4.1. Balance de datos

4.1.1. Clasificación de las supernovas clases SNIa, SNII, SLSN y SNIbc

Debido al gran desbalance de datos los cuales afectan directamente a los resultados, se plantea realizar un **Oversampling de las clases minoritarias (SMOTE)**, con el fin de obtener una base de datos balanceada. Siguiendo las recomendaciones del cuerpo docente, se agruparon los registros de la clase SNII, SNIIb y SNIIn en una sola clase. Debido a que hay una gran diferencia entre las clases, no se oversamplearon los datos hasta llegar al número de la clase mayoritaria pues esto era generar demasiados datos sintéticos.



(a) No balanceadas

(b) Balanceadas

Figura 2: Balance de datos clasificación multiclase con la clase II agrupadas.

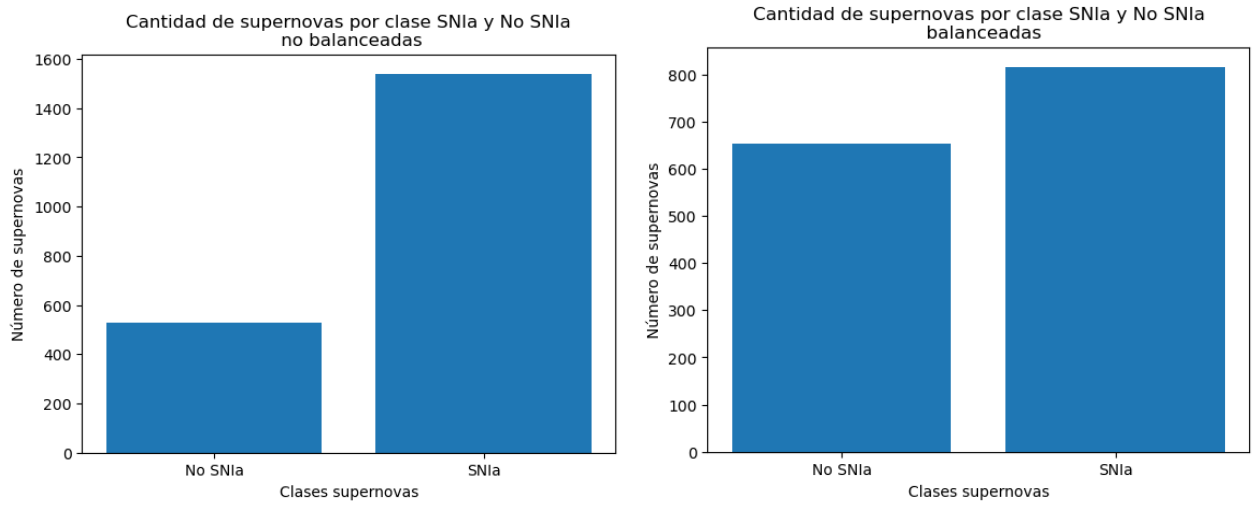
4.1.2. Clasificación binaria de las supernovas clases SNIa y No SNIa

Para esta clasificación se utilizaron solamente dos clases, "SNIa" (la clase mayoritaria) y "No SNIa", la cual engloba todos los demás tipos de supernova (SLSN, SNII, SNIIb, SNIIn, SNIbc). Con esto se ayuda a solucionar el problema del desbalance, pero no totalmente, por este motivo se debe usar igualmente el método denominado como *Synthetic Minority Oversampling Technique* o SMOTE. Este fue configurado como sigue:

Código 1: SMOTE para la clasificación binaria

```
1 Oversample = SMOTE(sampling_strategy = 0.8)
```

Donde el parámetro *sampling_strategy* define la razón entre el número de muestras en la clase minoritaria (No SNIa) luego del oversampling y el número de muestras en la clase mayoritaria (SNIa). Por lo tanto, considerando que luego del preprocesamiento de los datos se tienen 1166 muestras para la clase mayoritaria, al aplicar lo mostrado en el código 1, se aumenta la clase minoritaria a 933 muestras aproximadamente (se duplica).



(a) No balanceadas (b) Balanceadas

Figura 3: Balance de datos clasificación binaria

4.2. Conjunto de entrenamiento y test

Se utilizó la función train_test_split() de scikitlearn para realizar la separación de los conjuntos. Se utilizó el 70 % de los datos para el entrenamiento y el 30 % restante para test, no se utilizó un conjunto de validación debido a la poca cantidad de muestras por clases. Además de esto la partición se realizó utilizando el flag STRATIFY y el flag $shuffle = True$, esto para obtener un buen conjunto aleatorio para realizar tanto el entrenamiento como el test.

5. Algoritmos a Utilizar

Se realizaran clasificaciones con 2 diferentes algoritmos de clasificación supervisada, luego de familiarizarse con el problema y gracias a las experiencias que se han tenido tanto en clases como en las tareas del curso, se eligieron los siguientes 2 algoritmos:

1. **Random Forest (RF)**: Este algoritmo surge como evolución de los árboles de decisión y una modificación de bagging (*bootstrap aggregating*) y consiste principalmente en generar múltiples árboles de decisión descorrelacionados (se introduce aleatoriedad) sobre un conjunto de datos de entrenamiento. Este algoritmo mejora los árboles de decisión logrando un modelo más confiable y preciso, donde cada uno de los árboles creados por RF contiene un conjunto de observaciones aleatorias, las cuales son elegidas mediante bootstrap, que es una técnica para estimar la estadística de una población.
   Dentro del algoritmo, existen dos parámetros que son de suma importancia, el primero es el número de variables de entrada m, el cual debe elegirse tal que $m < p$, sin embargo, típicamente está establecido que para problemas de clasificación, $m = \sqrt{p}$ y el segundo corresponde a la cantidad de árboles que son generados con el algoritmo (sea N). El parámetro m, introduce aleatoriedad al modelo, lo que permite controlar la correlación entre los árboles. Por otro lado,

según la experimentación, a medida que N es mayor se producen menos problemas por *overfitting*. Este algoritmo posee un gran desempeño, ya que es una técnica eficiente para grandes bases de datos y es más simple que otras, como por ejemplo las RNA, además, es simple balancear las clases usando RF, lo cual facilita la aplicación en este problema en particular, donde como se pudo ver en secciones anteriores, las clases correspondientes a las supernovas están desbalanceadas. Por este motivo, es un algoritmo bastante usado generalmente en astronomía y el equipo lo implementará para clasificar supernovas. A continuación, se presenta un esquema del algoritmo para visualizar su comportamiento.

2. **Multilayer perceptron (MLP)**: Corresponde a una red neuronal artificial que tiene la capacidad de resolver problemas que no son necesariamente linealmente separables, esta consiste en un conjunto de capas de neuronas conectadas entre si, en donde existe una conjunto/capa de entrada, capas ocultas y una capa de salida. Cada una de las neuronas de la red posee pesos (y bias), una función de activación y un valor de error, el cual se trata de minimizar en cada iteración/mini-batch.
   Dentro de este algoritmo existen hiper-parámetros como la tasa de aprendizaje, el número de neuronas en la capa oculta, probabilidad de dropout, función de activación, etc.
   Este algoritmo posee un gran desempeño pues cuenta con regiones arbitrarias como fronteras de decisión, lo cual permite adaptarse y aproximar cualquier tipo de función. Se utilizan las redes MLP generalmente para clasificación, reconocimiento, recomendador, etc.

6. Software Utilizado

Se desarrolla el proyecto utilizando Python 3.8.8, usando como interfaz Jupyter notebook/Google Colab. Se utilizan librerías tales como numpy, pandas, Scikit-Learn, Scipy, imbalanced learn (aún en proceso), seaborn, entre otras.
Además de esto, se utilizan los extractores de características provistos por ALeRCE, en especifico la librería de **lc_classifier**, en donde se utilizaron los *Feature extractors* que provee esta librería.
El trabajo del equipo fue programar la sección de código que limpió los datos (tanto eliminar outliers, corrección de datos tipo NaN, entre otros) y la selección de conjuntos de entrenamiento y test, ya que, como fue mencionado al comienzo de la sección, para el entrenamiento y la extracción de características se utilizaron las librerías de Python y las provistas por el equipo docente.

7. Resultados

Para medir el rendimiento de los clasificadores se utilizaran las métricas de Balanced accuracy (se utilizará como B_accuracy en los resultados), Recall y Precision. Además, se utilizará distinto número de características para estas clasificaciones, primero se realizará con el total extraído (118) y luego con 46 seleccionadas según importancia que se muestran en la sección 12.2.

8. Clasificación utilizando Random Forests (RF)

8.1. Clasificación Binaria (SNIa, No SNIa)

Como fue mencionado en secciones anteriores, esta clasificación se realizó agrupando las clases minoritarias en una sola para realizar un modelo que se entrene en base a dos tipos de supernovas, "SNIa" y "No SNIa". El clasificador utilizado se muestra a continuación.

Código 2: Clasificador Random Forest

```
classifier_1 = RandomForestClassifier(
 n_estimators=50,
 criterion='gini',
 max_depth=30,
 max_features='sqrt',
 n_jobs=-1,
 class_weight='balanced'
)
```

Primero, se realizó una clasificación utilizando 118 características, tanto para los datos desbalanceados como para los datos luego del oversampling, con lo cual se obtuvieron las siguientes métricas de desempeño.

Tabla 2: Métricas de desempeño de clasificación binaria con 118 características

Clases	B_Accuracy [%]	Precision [%]	Recall [%]
Desbalanceadas	84.699134	90.778897	84.699134
Balanceadas	85.878788	89.511823	85.878788

En la tabla 2 se puede observar que luego del oversampling a los datos las métricas de desempeño mejoran, con lo cual el modelo es capaz de reconocer mejor la clase y con mayor precisión. Además de esto, a continuación se presentan las matrices de confusión (normalizadas) obtenidas para cada clasificación.

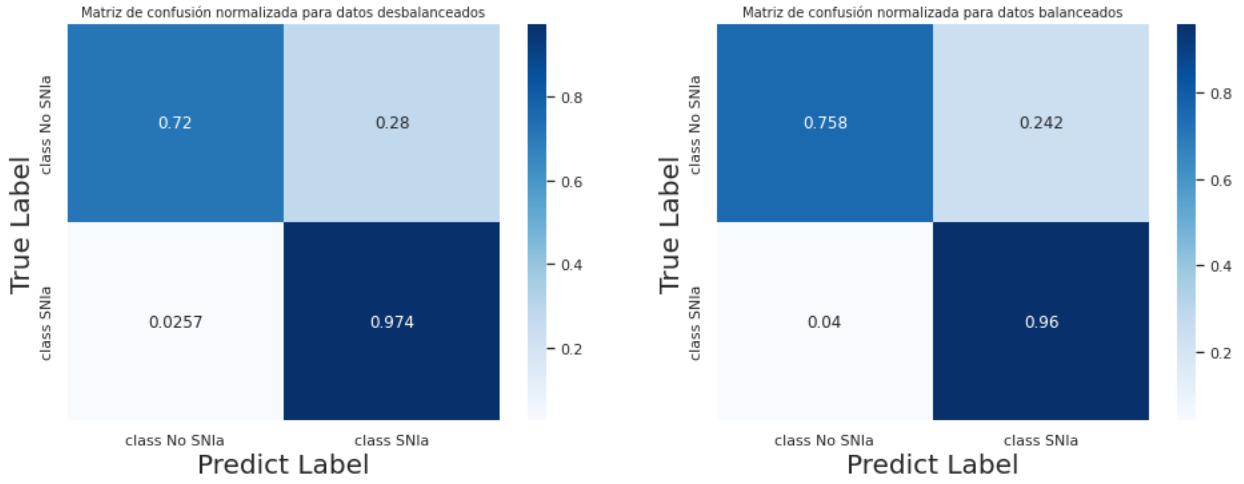


Figura 4: Matrices de confusión tanto de datos desbalanceados como balanceados

Como se puede observar en la figura 4 la clasificación es bastante eficiente y más aún con datos balanceados (figura de la derecha) donde se puede ver claramente la diagonal de la matriz con valores cercanos a uno (0.811 y 0.957).
Luego, se procedió a realizar una serie de cinco experimentos distintos para observar el promedio y la desviación de las métricas de desempeño obtenidas. Estos resultados se pueden ver a continuación.

Tabla 3: Resultados promedio obtenidos usando Random Forests, al utilizar 118 características.

Clases	**B_Accuracy [%]**	**Precision [%]**	**Recall [%]**
Desbalanceadas	82.22 ± 1.42	91.00 ± 1.39	82.22 ± 1.42
Balanceadas	84.56 ± 1.49	89.40 ± 1.62	84.56 ± 1.49

No obstante, ocupar 118 características puede ser un problema en términos de recursos computacionales en caso de tener muchas más muestras por clase. Por este motivo, se intenta disminuir la cantidad de características utilizadas para optimizar los recursos utilizados. Este procedimiento se puede ver en la siguiente sección.

8.1.1. Clasificación binaria seleccionando características importantes

Para esto, se tomaron los extractores de características proporcionados por el equipo docente y se realizó la extracción por separado, obteniendo 9 conjuntos distintos de características (es decir, uno por cada extractor). Con lo anterior se realiza el mismo procedimiento que para la clasificación con el conjunto completo, es decir, se unen las clases, se eliminan filas que tengan datos de tipo NaN y se realizan ambas clasificaciones (para datos desbalanceados y balanceados). Luego de esto, se utiliza la función *feature.importances_* de RandomForestClassifier para comparar las características que proporciona cada extractor y seleccionar sólo las más importantes. Con esto, se disminuye la cantidad de características utilizadas a 46, obteniéndose los siguientes resultados.

Tabla 4: Métricas de desempeño de clasificación binaria con 46 características

Clases	B_Accuracy [%]	Precision [%]	Recall [%]
Desbalanceadas	79.316796	90.463294	79.316796
Balanceadas	83.065929	92.300238	83.065929

Como se puede observar en la tabla anterior (4), los resultados con las clases balanceadas son un poco menores a los obtenidos con más del doble de características, pero el modelo sigue clasificando bien y obteniendo buenos resultados, por lo que se puede optar por usar menos características y optimizar recursos. En otras palabras existe un *trade off* entre la cantidad de características utilizadas, los recursos utilizados y los resultados obtenidos, puesto que a mayor cantidad de características, mejores fueron los resultados (para este experimento), pero los recursos utilizados aumentan también, lo cual puede ser contraproducente para otros casos. Finalmente, a continuación se presentan las matrices de confusión obtenidas para estas clasificaciones.

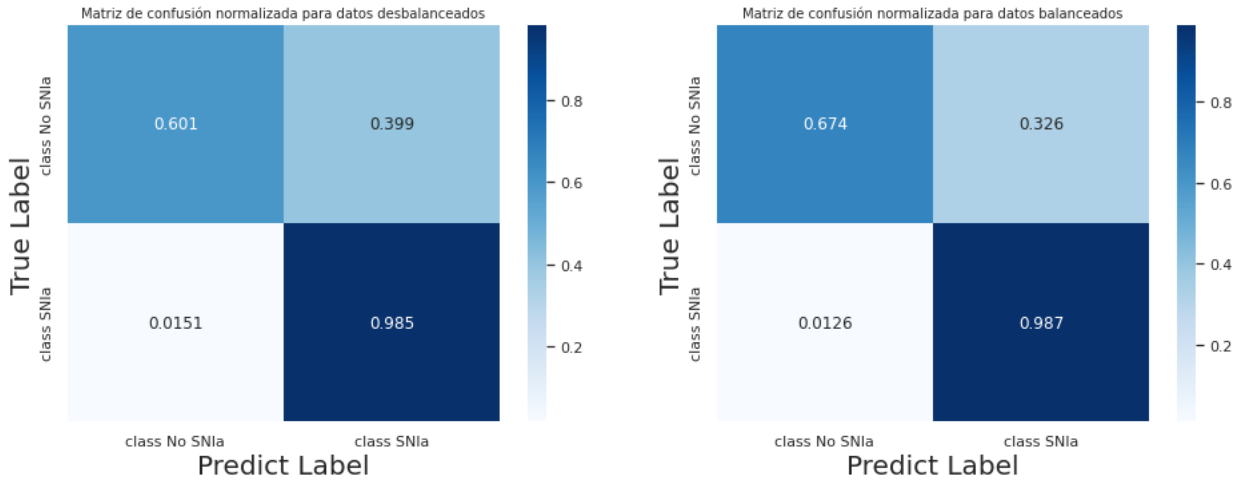


Figura 5: Matrices de confusión tanto de datos desbalanceados como balanceados

Al comparar las figuras 4 y 5 se puede observar que se "pierde" la diagonal de la matriz obteniendo valores de 0.674 y 0.987 para la clasificación con datos balanceados, pero que aún así el modelo logra clasificar entre los dos tipos de supernova y obtener buenos resultados.

Considerando lo anterior, nuevamente se procede a realizar una serie de cinco experimentos con la finalidad de comparar las métricas promedio del clasificador. Estos resultados se presentan a continuación.

Tabla 5: Resultados promedio obtenidos usando Random Forests, al utilizar 46 características.

Clases	B_Accuracy [%]	Precision [%]	Recall [%]
Desbalanceadas	80.28 ± 1.87	90.65 ± 0.62	80.28 ± 1.87
Balanceadas	83.78 ± 1.62	89.76 ± 1.00	83.78 ± 1.62

8.2. Clasificación multiclase (SLSN, SNII, SNIa, SNIbc)

Luego de los resultados eficientes obtenidos en la clasificación binaria, se procedió a realizar un modelo que pudiese clasificar entre los 4 tipos de supernova distintos. Este fue configurado de la misma manera que el mostrado en el código 3 y para analizar de mejor manera los resultados obtenidos se realizaron cinco experimentos, de los cuales se presenta a continuación el promedio y la desviación de cada métrica de desempeño y el mejor resultado para el clasificador Random Forests (entre estas cinco pruebas).
Primero, la tabla a continuación, resumirá el promedio y la desviación de las métricas de desempeño tanto para datos desbalanceados como balanceados.

Tabla 6: Resultados promedio obtenidos usando Random Forests, al utilizar 118 características.

Clases	B_Accuracy [%]	Precision [%]	Recall [%]
Desbalanceadas	43.67 ± 0.0	53.65 ± 0.0	43.67 ± 0.0
Balanceadas	51.6 ± 0.18	73.28 ± 4.27	51.6 ± 0.18

Luego de los cinco experimentos realizados, se rescata el mejor de ellos, cuyos resultados se muestran en la siguiente tabla.

Tabla 7: Mejor resultado obtenido usando Random Forests, al utilizar 118 características.

Clases	B_Accuracy [%]	Precision [%]	Recall [%]
Desbalanceadas	43.672547	53.653486	43.672547
Balanceadas	51.713673	79.519231	51.713673

Figura 6: Matriz de confusión normalizada para el mejor resultado, tanto de datos desbalanceados como balanceados.

8.2.1. Clasificación multiclase seleccionando características importantes

Al igual que en la sección anterior, se realizó una clasificación multiclase, pero esta vez se tomarán en cuenta sólo las 46 características más importantes para la clasificación. Nuevamente, se realizaron cinco experimentos cuyos resultados promedio se muestran a continuación con su respectiva desviación.

Tabla 8: Resultados promedio obtenidos usando Random Forests, al utilizar 46 características.

Clases	B_Accuracy [%]	Precision [%]	Recall [%]
Desbalanceadas	48.71 ± 0.0	68.82 ± 0.0	48.71 ± 0.0
Balanceadas	52.61 ± 0.38	83.22 ± 5.28	52.61 ± 0.38

Luego de los cinco experimentos realizados, se rescata el mejor de ellos, cuyos resultados se muestran en la siguiente tabla.

Tabla 9: Mejor resultado obtenido usando Random Forests, al utilizar 46 características.

Clases	B_Accuracy [%]	Precision [%]	Recall [%]
Desbalanceadas	48.714827	68.815055	48.714827
Balanceadas	52.264631	93.666424	52.264631

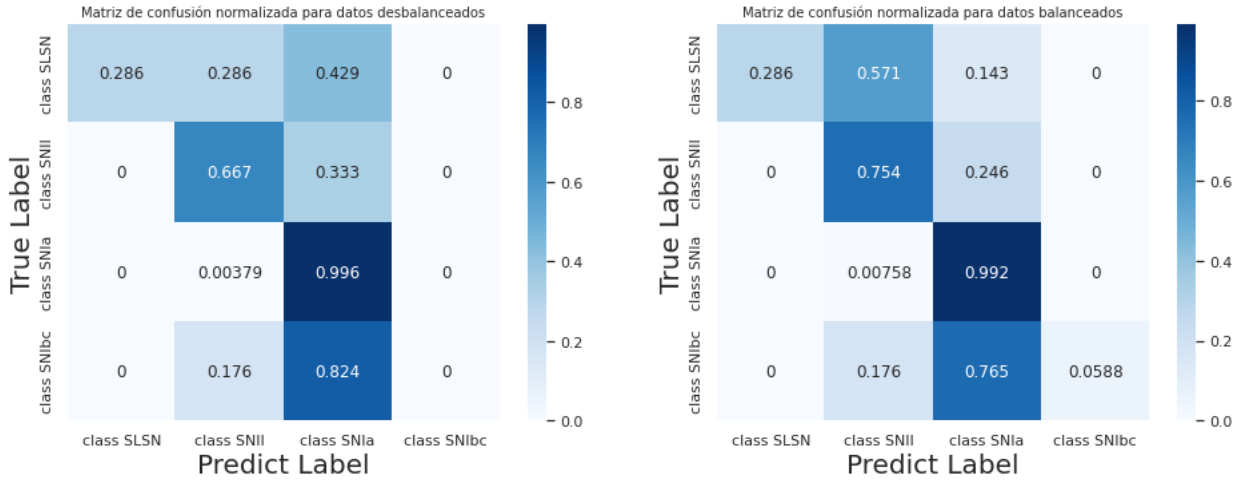


Figura 7: Matriz de confusión normalizada para el mejor resultado, tanto de datos desbalanceados como balanceados.

9. Clasificación utilizando Multilayer Perceptron (MLP)

La red utilizada para la clasificación de supernovas corresponde a la de la figura 8, el INPUT corresponde a las características/features utilizadas, luego posee varias capas ocultas donde se utiliza como función de activación sigmoides y relu's, entre las cuales hay capas de Dropout con el fin de evitar el overfitting y la capa de salida tiene una función de activación SOFTMAX con el fin de determinar a que clase es más probable que pertenezca según sus características. Esta red posee un **total de 40,388 parámetros** a entrenar. Se utilizó el algoritmo de optimización de **Adam**, una **tasa de aprendizaje de** $1e-3$, pues una mayor tasa de aprendizaje tenia una mayor inestabilidad en la convergencia y 'sparse_categorical_crossentropy' como función de costos.

Código 3: Configuración del modelo

```
# Configure the model and start training
model_smote.compile(loss='sparse_categorical_crossentropy',
optimizer=tf.keras.optimizers.Adam(learning_rate=1e-3), metrics=['accuracy'])
```

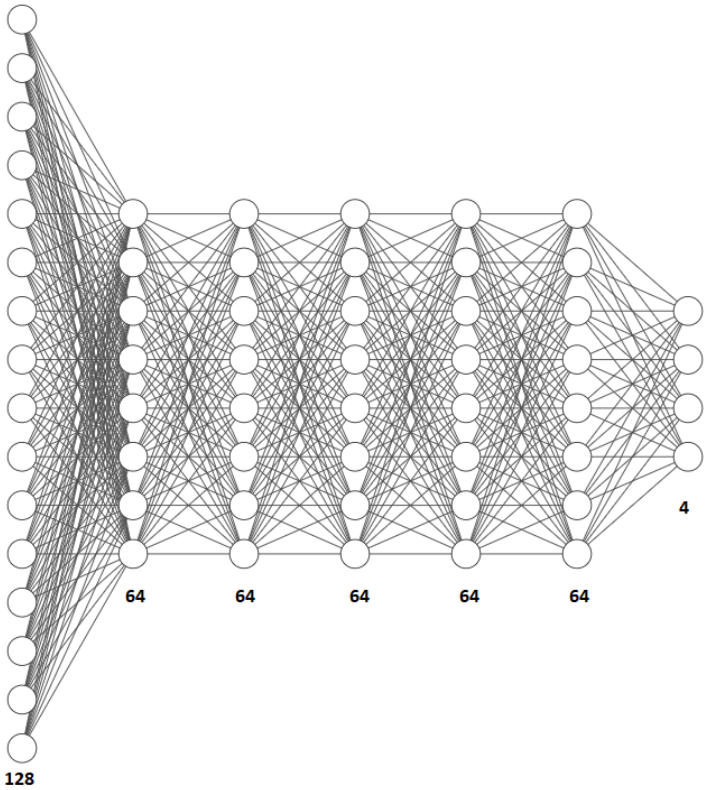


Figura 8: Estructura simplificada de la red utilizada, los números abajo de cada capa corresponden al número de neurona por capas. Esta red se puede observar en detalle en la figura 16.

9.1. Clasificación Binaria (SNIa, No SNIa)

Se realizó una clasificación utilizando 118 características, tanto para los datos desbalanceados como para los datos luego del oversampling, con lo cual se obtuvieron las siguientes métricas de desempeño.

Tabla 10: Métricas de desempeño de clasificación binaria con 118 características

Clases	B_Accuracy [%]	Precision [%]	Recall [%]
Desbalanceadas	83.11 +/- 1.0	87.43 +/- 0.65	83.11 +/- 1.0
Balanceadas	84.07 +/- 0.44	86.53 +/- 0.39	84.07 +/- 0.44

En la tabla 10 se puede observar que luego del oversampling a los datos, mejoran las métricas de desempeño, por otro lado al analizar las matrices de confusión de la figura 9, se obtienen mejores resultados de clasificación cuando se utilizan los datos oversampleados. A continuación se presentan las matrices de confusión (normalizadas) obtenidas para cada clasificación.

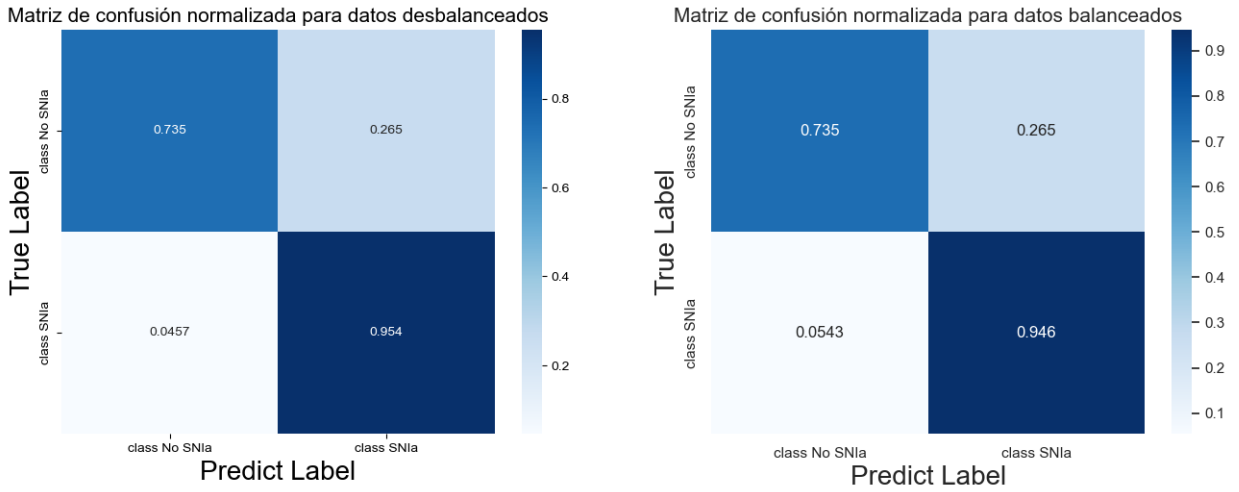


Figura 9: Matrices de confusión tanto de datos desbalanceados como balanceados utilizando 118 características. EL mejor modelo encontrado corresponde a la matriz de confusión balanceada, la cual presenta un 89.95 % de Accuracy y un 89.76 % de precision.

9.1.1. Clasificación binaria seleccionando características importantes

Al igual que en la sección de clasificación binaria utilizando el algoritmo Random Forest, se realizó la clasificación con las 46 características más importantes, según la función feature.importances_ de RandomForestClassifier para comparar las características que proporciona cada extractor y seleccionar sólo las más importantes.Se realizaron 5 experimentos para determinar que tan constantes eran los resultados de la red al utilizar estas 46 características, estos resultados se muestran en la tabla 11.

Tabla 11: Resultados promedio obtenidos usando la red MLP al utilizar las 46 características más importantes.

Clases	B_Accuracy [%]	Precision [%]	Recall [%]
Desbalanceadas	80.05 +/- 0.74	88.02 +/- 0.39	80.05 +/- 0.74
Balanceadas	84.27 +/- 1.09	87.04 +/- 0.93	84.27 +/- 1.09

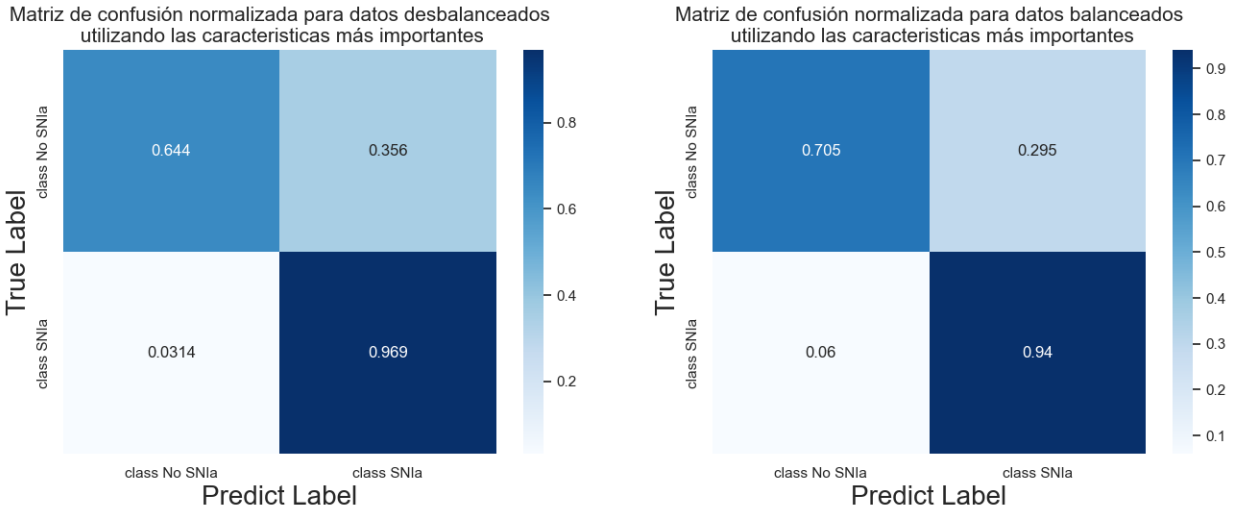


Figura 10: Matriz de confusión del mejor resultado, tanto de datos desbalanceados como balanceados al utilizar las 46 características más importantes.

Se puede observar en la figura 11 que el error en el conjunto de test va disminuyendo al aumentar las iteraciones convergiendo cerca de las 500 iteraciones, por otro lado el error en el conjunto de train tiende a cero.

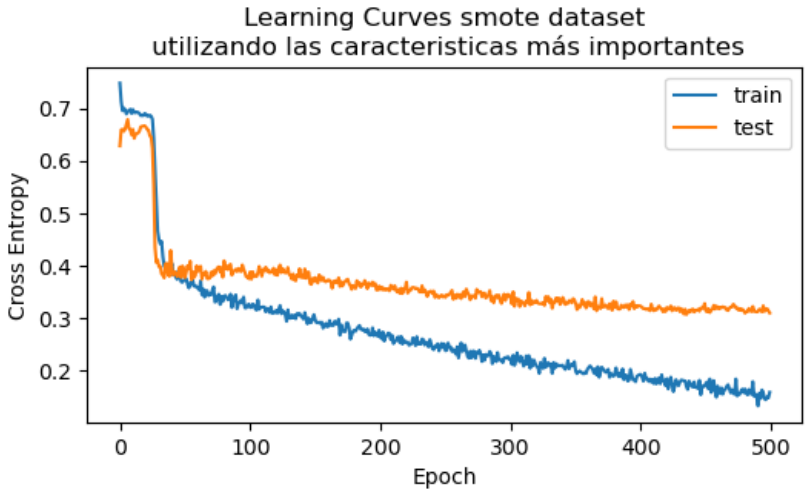


Figura 11: Curva de aprendizaje del mejor modelo usando los datos oversampleados al utilizar las 46 características más importantes.

9.2. Clasificación multiclase (SLSN, SNII, SNIa, SNIbc)

Se realizaron 5 experimentos para determinar que tan constantes eran los resultados de la red al utilizar 118 características, estos resultados se muestran en la tabla 14.

Tabla 12: Resultados promedio obtenidos usando la red MLP al utilizar 118 características.

Clases	B_Accuracy [%]	Precision [%]	Recall [%]
Desbalanceadas	48.64 +/- 4.57	52.25 +/- 9.53	48.64 +/- 4.57
Balanceadas	53.50 +/- 2.66	63.48 +/- 6.73	53.50 +/- 2.66

Tabla 13: Mejores resultados obtenidos usando la red MLP al utilizar 118 características.

Clases	B_Accuracy [%]	Precision [%]	Recall [%]
Desbalanceadas	51.39	58.91	51.39
Balanceadas	59.07	64.58	59.07

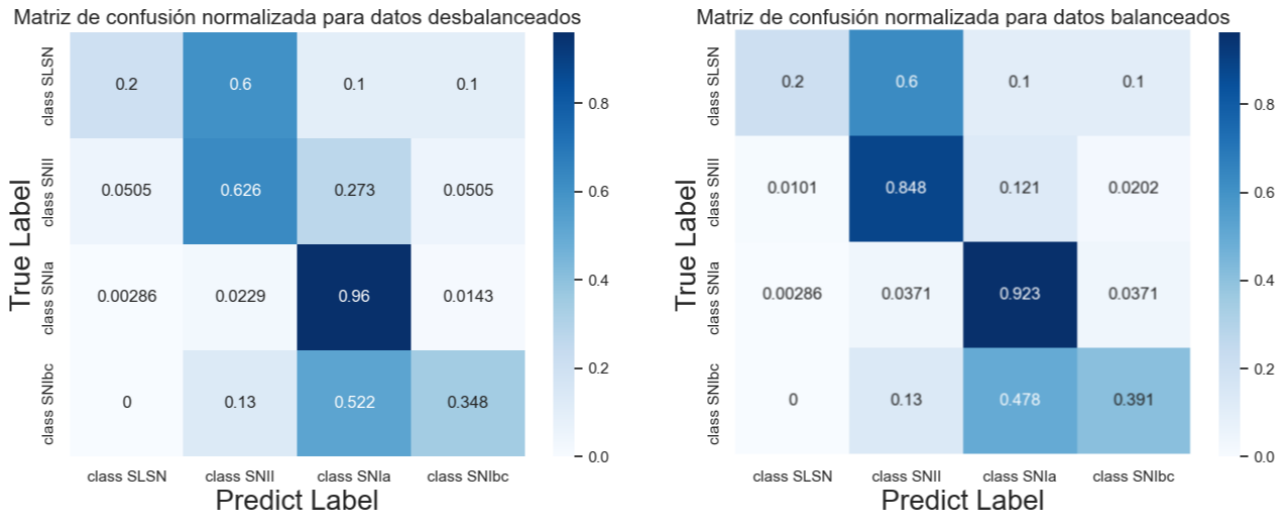


Figura 12: Matriz de confusión del mejor resultado, tanto de datos desbalanceados como balanceados.

Se puede observar en la figura 13 que el error en el conjunto de test va disminuyendo al aumentar las iteraciones convergiendo cerca de las 500 iteraciones, por otro lado el error en el conjunto de train tiende a cero.

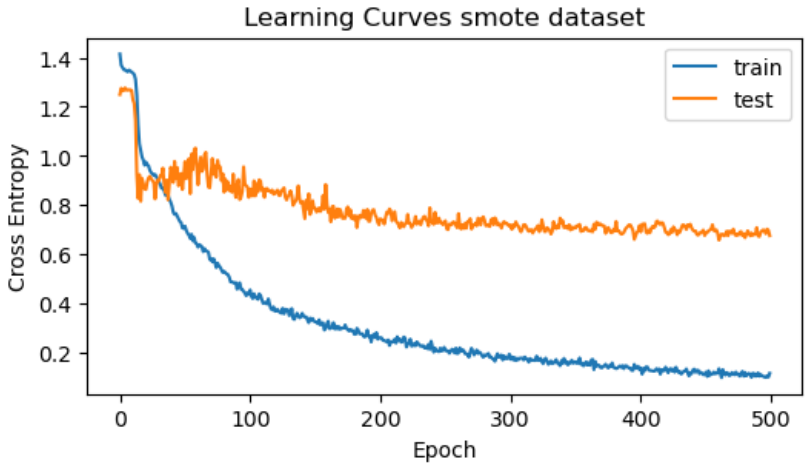


Figura 13: Curva de aprendizaje del mejor modelo usando los datos oversampleados.

9.2.1. Clasificación multiclase seleccionando características importantes

Al igual que en la sección de clasificación binaria utilizando el algoritmo Random Forest, se realizó la clasificación con las 46 características más importantes, según la función feature.importances_ de RandomForestClassifier para comparar las características que proporciona cada extractor y seleccionar sólo las más importantes.

Se realizaron 5 experimentos para determinar que tan constantes eran los resultados de la red al utilizar estas 46 características, estos resultados se muestran en la tabla 14.

Tabla 14: Resultados promedio obtenidos usando la red MLP al utilizar las 46 características más importantes.

Clases	**B_Accuracy [%]**	**Precision [%]**	**Recall [%]**
Desbalanceadas	49.26 +/- 3.21	49.79 +/- 3.83	49.26 +/- 3.21
Balanceadas	51.03 +/- 0.95	53.01 +/- 1.64	51.03 +/- 0.95

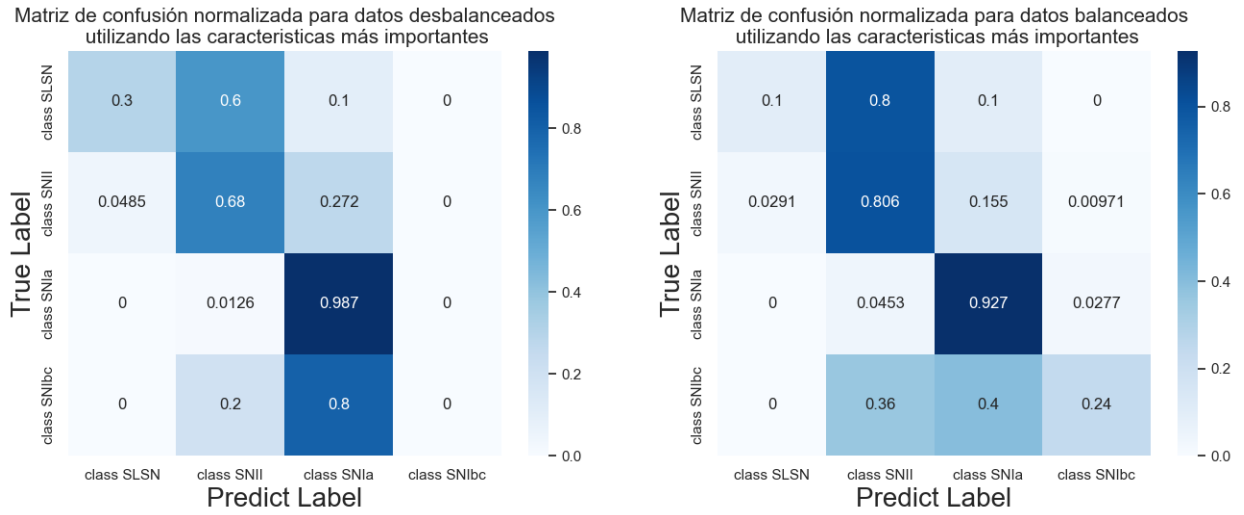


Figura 14: Matriz de confusión del mejor resultado, tanto de datos desbalanceados como balanceados al utilizar las 46 características más importantes.

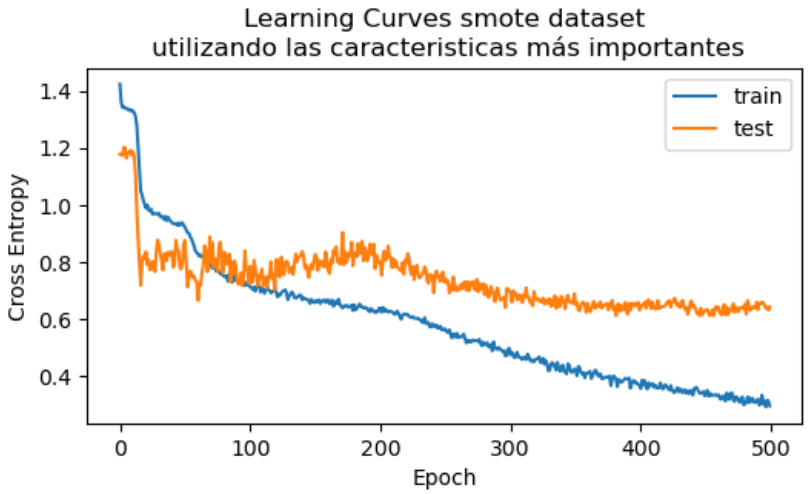


Figura 15: Curva de aprendizaje del mejor modelo usando los datos oversampleados al utilizar las 46 características más importantes.

10. Discusión

Observando los datos mostrados en la sección 7, se puede destacar lo siguiente.
Primero, el modelo creado tanto para Random Forests como para Multilayer Perceptron en clasificación binaria obtiene excelentes resultados en cuanto a exactitud y presición, exponiendo que ambos logran detectar bien entre una clase y otra, y lo realizan de manera confiable.
Luego, al observar los resultados de los modelos para clasificación multiclase se obtiene una alta precisión pero un bajo recall, manifestando que ambos no detectan de manera óptima la clase, pero que cuando lo detectan es confiable. Considerando esto, se procede a analizar el mejor modelo para cada tipo de clasificación (binaria o multiclase).

10.1. Selección y análisis el mejor modelo de clasificación Binaria de Supernovas (SNIa, No SNIa)

Observando los resultados de las tablas 3, 5, 10 y 11 se observa que el modelo que presenta mejores resultados en promedio y que presenta un error aceptable en sus métricas es el de la red MLP utilizando 118 características y datos balanceados (por oversampling). Este presenta tanto la exactitud balanceada, como la precisión y el recall sobre el 88 %, demostrando que el modelo puede detectar la clase de manera precisa y confiable, asegurando una buena clasificación. No obstante, debido a los parámetros de esta red y la cantidad de características utilizadas este requiere de más recursos computacionales que el modelo en base a Random Forests, el cual presenta sus métricas de desempeño sobre el 84 %. A continuación, se presenta una tabla resumen comparando el modelo tanto de Random Forests como de la red MLP, mostrando sus métricas de desempeño, tiempo que tarda en realizar la clasificación y número de características utilizadas.

Tabla 15: Tabla comparativa entre el mejor modelo RF y MLP

	Random Forests	**MLP**
Accuracy [%]	84.56 ± 1.49	88.15 ± 1.15
Precision [%]	89.40 ± 1.62	88.65 ± 0.71
Recall [%]	84.56 ± 1.49	88.15 ± 1.15
Tiempo [s]	2-5	180-300
Características	118	118

Al observar la tabla anterior (15), se puede nuevamente mencionar el *trade off* entre recursos y resultados. Luego, como las métricas de desempeño de ambos modelos están cercanas entre sí, se analiza la gran diferencia existente entre los tiempos que tarda el modelo en realizar la clasificación, donde se observa claramente que la red MLP tarda entre 15 y 60 veces más que el modelo de Random Forests, por lo tanto, para este problema en específico, con los datos presentados el modelo RF es el que mejor clasifica de forma binaria.

10.2. Selección y análisis el mejor modelo de clasificación multiclase de Supernovas (SLSN, SNII, SNIa, SNIbc)

Al comparar los mejores resultados de los modelos RF y MLP, lo cual se puede observar en la tabla 16, se obtiene un mejor acucuracy en promedio utilizando el modelo MLP, pero este tiene una mayor desviación en comparación con el modelo random forest, es decir el modelo random forest si bien tiene un Valor levemente menor de accuracy este es más robusto. Por otro lado al comparar la 'precision' de ambos modelos, el modelo random forest tiene un valor 20 % mayor en este indicador de rendimiento, al observar las matrices de confusión respectivas (figuras 7 y 12), es posible observar que el modelo random forest clasifica muy bien la clase SNIa (99.2 % de clasificaciones correctas), pero las demás clases baja bastante las clasificaciones correctas (28.6 % ,75.4 % y 5.88 % de clasificaciones correctas en las clases SLSN, SNII y SNIbc respectivamente). Por otro lado la red MLP clasifica un poco peor la clase SNIa (92.3 % de clasificaciones correctas), pero clasifica mejor todas las clases en general (20 %, 84.8 % y 39.1 % de clasificaciones correctas en las clases SLSN, SNII y SNIbc respectivamente). Considerando esto y los resultados obtenidos en la tabla 16, el mejor modelo corresponde al modelo MLP utilizando 118 características, esto pues a pesar de requerir un mayor costo computacional y de tiempo, es el que cumple de mejor forma el objetivo de clasificar las Supernovas SLSN, SNII, SNIa, SNIbc.

Tabla 16: Tabla comparativa entre el mejor modelo RF y MLP clasificación mutliclase.

	Random Forests	**MLP**
Accuracy [%]	52.61 ± 0.38	53.50 ± 2.66
Precision [%]	83.22 ± 5.28	63.48 ± 6.73
Recall [%]	52.61 ± 0.38	53.50 ± 2.66
Tiempo [s]	2-5	180-300
Características	46	118

11. Conclusión

A partir de los resultados obtenidos se puede observar que el objetivo principal del proyecto fue cumplido, esto debido a que se logró realizar un modelo supervisado que lograra clasificar las supernovas entre sus distintas clases, no obstante, este no obtuvo los resultados deseados en cuanto a métricas de desempeño (para clasificación multiclase), por lo que se presenta un gran déficit en esta área y los resultados de la literatura en esta clasificación son mejores que los logrados con le modelo realizado. Sin embargo, el modelo creado para clasificación binaria obtuvo excelentes resultados, logrando detectar las clases de manera correcta y confiable. Además de esto, se realizó un análisis post-clasificación de las mejores características según el algoritmo Random Forests las cuales se presentan en el anexo 12.2.

Este proyecto le permitió al equipo conocer la importancia de las supernovas en la observación astronómica y en el estudio de los fenómenos del espacio, además de familiarizarse con dos algoritmos de aprendizaje supervisado importantes en el campo como lo es Random Forests y Multilayer Perceptron. Este aprendizaje, trajo consigo una serie de dificultades a lo largo del proyecto, como fue la primera extracción de características utilizando los extractores provistos por el equipo docente y la primera implementación de un clasificador, en conjunto con los bajos resultados obtenidos para la clasificación multiclase. Se estima que esta última dificultad sea solucionable mediante un estudio más a fondo de las características y de técnicas de balanceo de datos, permitiendo que el modelo se entrene de manera equitativa entre los ejemplos de cada clase y clasifique eficientemente los tipos de supernovas presentes.

Cabe destacar que luego de los comentarios realizados por el equipo docente en la presentación final, el equipo probó con un nuevo clasificador de la librería *imblearn* denominado *"BalancedRandomForestClassifier"*, el cual a simple vista presentó mejores resultados para la clasificación multiclase en cuanto a métricas de desempeño y a su matriz de confusión, por lo tanto, también es una excelente alternativa para mejorar los resultados del proyecto.

Referencias

[1] Sánchez-Sáez, P., et al. .Alert Classification for the ALeRCE Broker System: The Light Curve Classifier..arXiv preprint arXiv:2008.03311 (2020).

[2] Rocío Espinar Lara. Métodos de clasificación con datos no balanceados. Facultad de matemáticas, Universidad de Sevilla. Junio 2018. páginas 47 y 48. https://idus.us.es/bitstream/handle/11441/77518/Espinar%20Lara%20Roc%C3%ADo%20TFG.pdf?sequence=1&isAllowed=y

[3] Javier Jesús Espinosa-Zúñiga, Aplicación de algoritmos Random Forest y XGBoost en una base de solicitudes de tarjetas de crédito, revista Scielo, Ing. invest. y tecnol. vol.21 no.3, 2020. Disponible en: http://www.scielo.org.mx/scielo.php?pid=S1405-77432020000300002&script=sci_arttext&tlng=es

[4] Marta García Ruiz de León, Análisis de Sensibilidad Mediante Random Forest, Universidad Politécnica de Madrid, 2018, páginas 4 y 5. Disponible en: https://oa.upm.es/53368/1/TFG_MARTA_GARCIA_RUIZ_DE_LEON.pdf

[5] J. Torres, DEEP LEARNING: INTRODUCCIÓN PRÁCTICA CON KERAS, Primera parte. Junio 2018 Disponible en: https://torres.ai/deep-learning-inteligencia-artificial-keras/

[6] Scikit-learn, User Guide. 2021 Disponible en: https://scikit-learn.org/stable/user_guide.html

12. Anexos

12.1. Modelo MLP utilizado

```
Model: "mlp_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_89 (Dense)             (None, 128)               15232
_________________________________________________________________
dropout_59 (Dropout)         (None, 128)               0
_________________________________________________________________
dense_90 (Dense)             (None, 64)                8256
_________________________________________________________________
dropout_60 (Dropout)         (None, 64)                0
_________________________________________________________________
dense_91 (Dense)             (None, 64)                4160
_________________________________________________________________
dropout_61 (Dropout)         (None, 64)                0
_________________________________________________________________
dense_92 (Dense)             (None, 64)                4160
_________________________________________________________________
dropout_62 (Dropout)         (None, 64)                0
_________________________________________________________________
dense_93 (Dense)             (None, 64)                4160
_________________________________________________________________
dropout_63 (Dropout)         (None, 64)                0
_________________________________________________________________
dense_94 (Dense)             (None, 64)                4160
_________________________________________________________________
dense_95 (Dense)             (None, 4)                 260
=================================================================
Total params: 40,388
Trainable params: 40,388
Non-trainable params: 0
```

Figura 16: Estructura de la red utilizada.

12.2. Características utilizadas en clasificación

Este apartado contiene el listado de las 46 características utilizadas en la clasificación. Cabe destacar, que estas no están ordenadas según importancia, pero que en general, las más importantes son las correspondientes al extractor SNParametricModelExtractor (comienzan por SPM).

1. delta_mag_fid_1
2. delta_mag_fid_2
3. Multiband_period
4. PPE
5. Period_band_2
6. delta_period_2
7. MHPS_ratio_1
8. MHPS_low_1
9. MHPS_high_1
10. MHPS_low_2
11. MHPS_high_2
12. GP_DRW_sigma_1
13. GP_DRW_tau_1
14. GP_DRW_sigma_2
15. GP_DRW_tau_2
16. Psi_CS_1
17. Psi_eta_1
18. Psi_CS_2
19. Psi_eta_2
20. Harmonics_phase_2_1
21. Harmonics_phase_3_1
22. Harmonics_phase_4_1
23. Harmonics_mag_2_2
24. Harmonics_phase_3_2
25. Harmonics_phase_7_2
26. Harmonics_mse_2
27. Power_rate_1/4
28. Power_rate_1/3
29. Power_rate_1/2
30. Power_rate_2
31. Power_rate_3
32. Power_rate_4
33. IAR_phi_2
34. LinearTrend_2
35. SPM_A_1
36. SPM_t0_1
37. SPM_gamma_1
38. SPM_beta_1
39. SPM_tau_rise_1
40. SPM_tau_fall_1
41. SPM_A_2
42. SPM_t0_2
43. SPM_gamma_2
44. SPM_beta_2
45. SPM_tau_rise_2
46. SPM_tau_fall_2