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

Informe Preliminar

Clasificación de Supernovas de ALeRCE - ZTF

Integrantes: Salvador Mercado
Pablo Duarte
Profesor: Pablo Estevez
Auxiliar: Pablo Cornejo
Ayudantes: Bruno Araya
Diego Castillo
Francisco Soto
Jorge Espejo
Rodrigo Catalán
Sebastian Guzman

Fecha de realización: 26 de octubre de 2025
Fecha de entrega: 26 de octubre de 2025
Santiago de Chile

Índice de Contenidos

Índice de Figuras

Índice de Tablas

1. Descripción del Problema y Motivación

En la actualidad la astronomía ha entrado en la era de los sondeos de gran etendeu (como ZTF (Zwicky Transient Facility) y en el futuro el Observatorio Vera C. Rubin, LSST). Estos telescopios escanean el cielo continuamente, generando grandes volúmenes de "alertas" de eventos astronómicos (millones de eventos por noche) que señalan cambios en el brillo o posición de los objetos.

Dado el gran volumen de alertas por noche emitidas (ZTF genera entre 10^5 y 10^6 alertas nocturnas, y se esperan 10^7 para LSST), ningún equipo humano puede procesar esta cantidad de datos. Esta avalancha de información ha hecho necesaria la creación de "brokers" de alertas: sistemas automatizados que ingieren, procesan y filtran este flujo de datos para la comunidad.

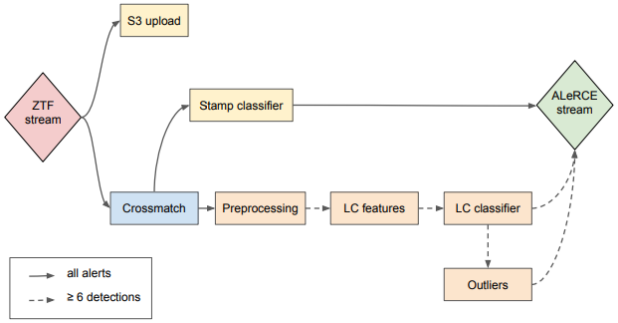


Figura 1: Arquitectura y Pipeline del Broker de ALeRCE.

Es por ello que ALeRCE (Automatic Learning for the Rapid Classification of Events), un broker de origen chileno, fue diseñado para recibir y procesar el flujo masivo de alertas en tiempo real de ZTF y, próximamente, también podrá recibir los datos de LSST.

El sistema ALeRCE utiliza un pipeline en tiempo real que ingiere las alertas de ZTF (distribuidas vía Apache Kafka) y las procesa en varios pasos:

- **Ingesta y Agregación:** Recibe la alerta y la agrega a la información existente del objeto.
- **Cross-matching:** Cruza la alerta con catálogos externos (como WISE, PanSTARRS, Gaia) para obtener más información.
- **Clasificación de ML:** Aplica sus modelos de machine learning.
- **Visualización y Distribución:** Pone los resultados a disposición del público a través de herramientas web y APIs.

El núcleo de este pipeline son sus clasificadores de machine learning, diseñados para operar en diferentes etapas11. ALeRCE utiliza un Clasificador de Estampillas (basado en imágenes) para una clasificación rápida (p.ej., identificar Supernovas en su primera alerta), y un Clasificador de Curvas de Luz (LCC) para una clasificación refinada una vez que se han acumulado suficientes datos (definido como ≥ 6 detecciones).

Este proyecto se sitúa en el dominio del segundo clasificador, abordando uno de los desafíos científicos clave de ALeRCE: la clasificación detallada de fenómenos transitorios.

Si bien la primera versión del LCC de ALeRCE (descrita en Sánchez-Sáez et al. 2020) ya clasifica supernovas en cuatro clases generales (SNIa, SNIbc, SNII y SLSN), los propios autores señalan las dificultades inherentes a esta tarea. Existe una "confusión significativa" entre las clases de supernovas, particularmente entre SNIa y SNIbc, y entre SNII y SLSN. Además, esa taxonomía inicial agrupa subtipos importantes (como SNII y SNIIn) y omite explícitamente otros (como SNIIb) debido a la complejidad y la escasez de muestras de entrenamiento en ese momento.

Por lo tanto, este proyecto busca abordar directamente esta limitación. El objetivo es desarrollar un clasificador supervisado especializado, enfocado exclusivamente en supernovas, utilizando un conjunto de features pre-calculadas de sus series de tiempo (curvas de luz). Estas curvas de luz entregan información sobre el brillo estelar, su error y el día en que ocurrió.

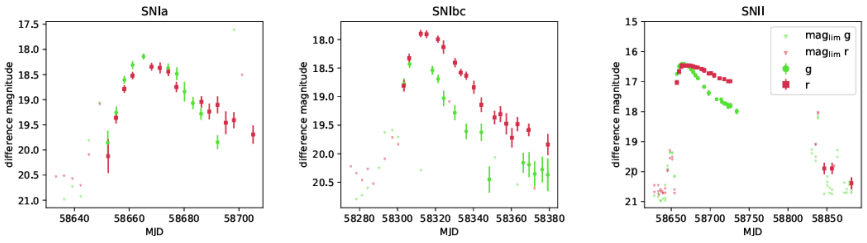


Figura 2: Curvas de Luz de tres tipos de Supernovas transientes.

2. Objetivos del Proyecto

- Inspeccionar, visualizar y preprocesar los datos de ALeRCE.
- Proponer un modelo supervisado para clasificar supernovas a partir de sus características de curva de luz en seis clases desbalanceadas.
- Estudiar la relevancia de cada característica de las curvas de luz en la clasificación final.
- Diseñar e implementar red MLP con Multi-classSVDD.
- Investigar e implementar técnicas de proyección/reducción de dimensionalidad (UMAP y t-SNE) de los vectores de características para realizar clustering.

- Inspección e identificación visual de outaliers.

3. Base de Datos

Para la realización del proyecto se disponen de tres tablas (o conjuntos de tablas) que conforman la base de datos principal para el entrenamiento y testeo de los modelos. A continuación se describen las características de cada una.

- **Tabla de Detecciones ZTF:** Esta tabla conceptual representa los datos de observación crudos procesados por ALeRCE. El pipeline ingiere los paquetes de alertas Avro de ZTF. Estos paquetes originales se almacenan en AWS S3 , mientras que la información fotométrica relevante se extrae y almacena en tablas de una base de datos PostgreSQL. Las tablas principales que componen esta base de datos son:
  - *detections:* Contiene las curvas de luz (series de tiempo), incluyendo las magnitudes de diferencia (m_{diff}) y las magnitudes aparentes corregidas (m_{corr}) calculadas por ALeRCE, junto con sus errores.
  - *non_detections:* Almacena los límites superiores de magnitud de las observaciones donde el objeto no fue detectado en los 30 días previos. Esta información es vital para caracterizar transitorios "nuevos".
  - *objects:* Contiene estadísticas agregadas por objeto, como la localización (RA, Dec) y las fechas de primera y última detección.
- **Tabla de Crossmatch:** Esta base de datos representa el "Conjunto Etiquetado" (Labeled Set), fundamental para entrenar y probar los modelos de machine learning. Se utiliza para construir el clasificador.

  Su construcción se realiza a partir de objetos observados por ZTF para los cuales ya se conoce su clasifiación astrofísica. Para enriquecer la base de datos con etiquetas confiables, ALeRCE cruza (crossmatch) las fuentes de ZTF con múltiples catálogos astronómicos públicos.

  Los catálogos específicos utilizados para construir el conjunto de etiquetas del Clasificador de Curvas de Luz (LCC) incluyen:
  - **Transitorios (SNe):** Transient Name Server (TNS).
  - **Estocásticos (AGN, QSO, Blazar, YSO, CV/Nova):** Million Quasars Catalog (MILLIQUAS), ROMABZCAT, New Catalog of Type 1 AGNs (Oh2015), SIMBAD y compilaciones de JAbril.
  - **Periódicos (Estrellas Variables):** Catalina Surveys (CRTS), ASAS-SN, LINEAR y Gaia DR2.

Este conjunto de datos solo incluye objetos que alcanzan un mínimo de 6 detecciones en banda g o 6 detecciones en banda r. Como se menciona en el enunciado del proyecto, este conjunto de etiquetas presenta un alto desbalance entre clases, un desafío que ALeRCE aborda usando algoritmos específicos como Balanced Random Forest.

- **Tabla de Características de Curva de Luz de Supernovas:** Esta es la base de datos procesada que se utiliza como entrada (input) directa para los modelos de machine learning. En lugar de usar las curvas de luz crudas, ALeRCE calcula un vector de 152 características (features) que resumen las propiedades de la serie de tiempo y del objeto.

  Estas features se almacenan en la tabla features de la base de datos (a menudo como archivos JSON) y solo se calculan cuando un objeto alcanza el umbral de ≥ 6 detecciones. El vector de 152 features se compone de:

  - **Características de Detección (124):** Se calculan a partir de las curvas de luz g y r. Incluyen estadísticas simples (de la librería FATS), modelos de ajuste (como el **SPM** o Supernova Parametric Model, crucial para este proyecto)y características novedosas como **IAR** (autocorrelación).
  - **Características de No-Detección (18):** Novedosas de ALeRCE, calculadas a partir de la tabla *non_detections*. Son cruciales para transitorios, midiendo, por ejemplo, el número de no-detecciones antes de la primera luz ($n\_non\_det\_before\_fid$).
  - **Metadatos y Colores (10):** Incluyen la morfología (sgscore1), coordenadas galácticas y colores infrarrojos de **AllWISE** (ej. $W1 - W2$).

4. Pre-procesamiento de Datos

Resumen de las dimensiones de las tablas antes del procesamiento de datos.

Tabla 1: Tabla Resumen de Dimensiones de las Bases de Datos del Proyecto.

Tabla	**Dimensiones**
Detecciones ZTF	128718 filas × 32 columnas
Crosmatch (Labeled Set)	173879 filas × 9 columnas
Características de Curva de Luz de Supernovas	1823 filas × 179 columnas

Como la base de datos de interés para el proyecto es la que contiene las características de curva de luz de las supernovas, se realizó el siguiente pre-procesamiento de los datos:

4.1. Selección y Transformación de Variables

El primer paso del preprocesamiento consiste en la validación de la integridad y el formato de la tabla de características (lc_feat).

Inicialmente, se comprobó que la tabla contenga objetos únicos, verificando que no existan identificadores duplicados en la columna *oid*. El *oid* (Object ID) es el identificador único que ALeRCE asigna a cada fuente astronómica detectada por ZTF. Paralelamente, se verificó que todas las columnas de características tuvieran un tipo de dato numérico uniforme (*float64*), esencial para el entrenamiento de los modelos. En ambos casos, los resultados fueron positivos y no fue necesario realizar correcciones.

A continuación, se revisó la existencia de valores nulos (NaN), contabilizando un total de 34332 valores nulos. Estos valores faltantes suelen originarse cuando una característica no se puede calcular; por ejemplo, si un objeto solo tiene suficientes detecciones en una de las dos bandas (g o r), las 56 características de la banda faltante aparecerán como nulos.

Siguiendo la metodología implementada por el equipo de ALeRCE en el clasificador de curvas de luz, todos los valores NaN se imputaron con el valor estático -999. Esta estrategia se adopta para mantener la consistencia del dataset y permitir que los algoritmos de machine learning (como Random Forest o MLP) procesen los datos, asumiendo que este valor extremo permitirá al modelo aprender a identificar y tratar la ausencia de información como una característica en sí misma, sin sesgar la distribución de las características existentes.

```
Objetos únicos: True
DataFrame float64: True
Valores nulos: 34332
```

Figura 3: Verificación de objetos únicos, tipo de datos y valores NaN en la Tabla de Características de Curvas de Luz de Supernovas.

4.2. Normalización

Una vez validada la calidad de la tabla de características, el siguiente paso fue la asignación de las etiquetas de "verdad fundamental" (ground truth).

Se realizó una operación $left\ merge$ (unión por la izquierda) entre la tabla de características (lc_feat) y la tabla de $Crossmatch$ (que contiene el conjunto etiquetado). La unión se realizó utilizando la columna *oid* como llave primaria. El resultado es un DataFrame que contiene todas las características originales de lc_feat y una nueva columna, $classALeRCE$, con la etiqueta de clase proveniente de la tabla $Crossmatch$.

Posteriormente, se realizó una transformación de las clases de interés para agrupar subtipos de supernovas y reducir la complejidad del problema. La taxonomía del LCC de ALeRCE original ya agrupa varias clases de supernovas (p.ej., SNe II y IIn) bajo la etiqueta genérica "SNII". Siguiendo un enfoque similar, las clases SNII, SNIIb y SNIIn (que comparten un

origen común de colapso de núcleo) se unificaron bajo una sola etiqueta.

La codificación numérica final de las clases para el entrenamiento es la siguiente:

Tabla 2: Tabla Resumen de Transformaciones de Clases a Representación Numérica.

Clase	**Representación Numérica**
SNIa	0
SNIbc	1
SNII	2
SNIIb	2
SNIIn	2
SLSN	3

4.3. Manejo de Desbalance de Clases

El conjunto de datos etiquetado (*Labeled Set*) utilizado por ALeRCE es conocido por sufrir un alto desbalance entre clases. Este es uno de los principales desafíos en el desarrollo de los clasificadores, ya que los modelos tienden a sesgarse hacia la clase mayoritaria e ignorar a las minoritarias.

Para mitigar este problema, especialmente en modelos sensibles al desbalance como las Redes Neuronales (MLP), se aplicaron técnicas de remuestreo. Este enfoque es consistente con las pruebas realizadas por el equipo de ALeRCE, quienes también exploraron el uso de $RandomUnderSampler$ y $RandomOverSampler$ (de la librería $imbalanced-learn$) para balancear los datos al probar arquitecturas GBoost y MLP.

El procesamiento realizado en este proyecto consistió en:

- Undersampling (submuestreo) de la clase mayoritaria (SNIa).
- Oversampling (sobremuestreo) de las clases minoritarias (SNIbc y SLSN).

5. Definición y Justificación de los Algoritmos Usados

En lo relativo a la selección de algoritmos predictivos, se exploraron configuraciones tanto con MLP y BRF (Balanced Random Forest). La selección de Balanced Random Forest tiene su justificación dada mayor robustez relativa a otros modelos ante ruido, sus fronteras de decisión rectangulares y criterios de entropía/gini y undersampling de clase mayoritaria nativa -generalmente dando buenos resultados en tareas de clasificación desbalanceada-. Cabe

mencionar que Balanced Random Forest fue uno de los clasificadores (junto con GBoost) que fueron más efectivos en el paper 'Alert Classification for the ALeRCE Broker System: The Light Curve Classifer'.

Por otro lado, si bien MLP no es teóricamente el modelo adecuado para analizar estos datos dados distinos tamaños de clase, vulnerabilidad ante ruido cuando no es parte de un ensamblado de varios modelos y fronteras de decisión inadecuadas para clases con solapamiento de atributos, se explora el uso de este modelo con el objetivo de analizarlo para implementación futura del algoritmo MulticlassSVDD, el cual si es exitoso catalogará con precisión las clases en mapeados esféricos, además de identificar efectivamente outliers en la distribución de los datos.

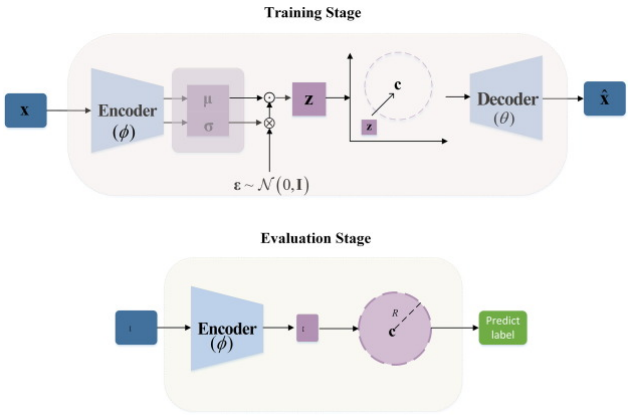


Figura 4: Visualización MLP con uso de encoder para MulticlassSVDD

En añadidura, para todos los modelos se implementó una visualización UMAP sobre el conjunto de prueba con `n_neighbours` = 30 y `min_dist` = 0.3, de manera de entender de manera más explicita como los clasificadores van agrupando y/o clasificando los datos, preservando estructura local de estos y por consecuencia una visión fidedigna del comportamiento de los modelos.

6. Configuración de modelos e hiperparámetros

En los siguientes items se explorarán detalles de selección de épocas, capas ocultas, estimadores, atributos a utilizar, funciones de perdida, criterios de learning rate y early stopping, además de fallas y aspectos aún no explorados.

6.1. Salidas Deseadas y Función Objetivo

Respecto a Balanced Random Forest, su output es una clase predicha para cada ejemplo, tomando el modelo de `imbalanced_learn` un voto sobre las predicciones discretas de cada

estimador como lo hace comunmente Random Forest estandar.

En contraste, la salida deseada para MLP es un vector de tamaño 4 con los logits, es decir, salidas de 4 probabilidades que siempre suman 1 (SNIa, SNIIbc, SNII, SLSN), de tal forma que si las probabilidades son z_i para cada clase i, estas pasen a ser $P(y_i) = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$. De esto se desprendería que la función objetivo será una de entropía cruzada con softmax multiclase, sin embargo dado diferencia en tamaño de clases y uso de UMAP se opto por una opción más adecuada para el problema, ocupando la suma de center loss y focal loss, cuyas fórmulas son presentadas a continuación.

$$\mathcal{L}_{\text{focal}} = -\frac{1}{N} \sum_{i=1}^{N} \alpha_{y_i} (1 - p_{y_i})^{\gamma} \log(p_{y_i})$$

$$\mathcal{L}_{\text{center}} = \frac{\lambda_c}{2N} \sum_{i=1}^{N} \|\mathbf{f}_i - \mathbf{c}_{y_i}\|_2^2$$

Al analizar las funciones presentadas -tomando en cuenta que α_{y_i} es un ponderador que crece en función de que tan pequeño es el porcentaje de datos de la clase y_i, y que para center loss se asigna un centro c_i a cada embedding f_i del MLP (vector resultante antes de paso por la última capa) es posible concluir que focal loss penaliza mucho las clases minoritarias mal etiquetadas, mientras que center loss ayuda a la representación tanto para clasificación como para visualización con UMAP. Cabe mencionar que softmax es utilizado dentro de entropía cruzada para obtener los p_{y_i} de cada ejemplar mediante entropía cruzada, aunque esta última no es criterio principal.

6.2. Principio de Optimización Usado

Para BRF se utiliza el típico método de bagging para añadir variabilidad a las predicciones de los distintos estimadores. De manera similar en cuanto a estándares, para MLP se incorpora metodo Adam para descenso adaptativo de gradiente.

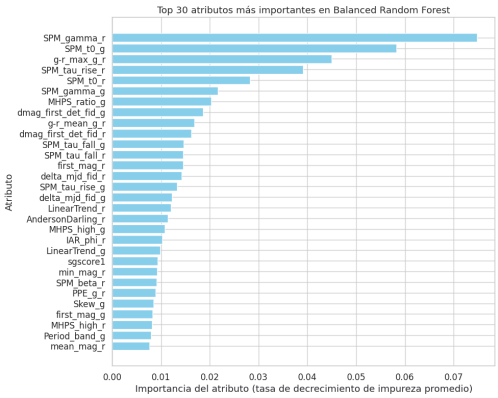
6.3. Criterio de Detención del Algoritmo

No se incorporo un criterio de detención ni para BRF o MLP, lo que deberá ser incorporado en futuras iteraciones del código final. Especialmente para MLP, el overfitting puede llegar a ser un problema grave, por lo que se agregara un mecanismo de early stopping en un futuro.

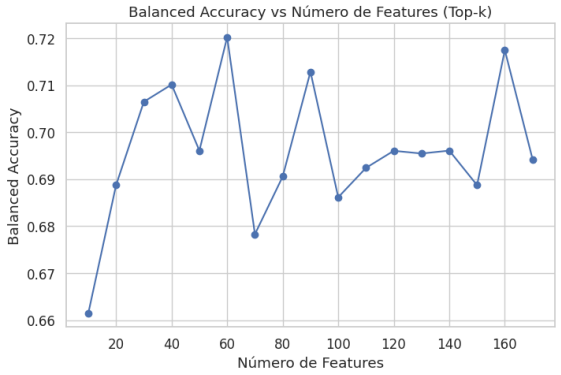
6.4. Parámetros por Definir y/o Ajustar

Indagando en la elección de parámetros para BRF, se efectuó un proceso de búsqueda de hiperparámetros óptimos para todas las opciones del modelo, es más, utilizando importancia de features ordenadas, se buscó mejorar la calidad de predicciones para cada inclusión de top 170 a top 10 atributos (pasos de -10). Tanto las características más importantes como el

hyperparameter tuning son mostradas a continuación.



(a) Importancia de las 30 características principales.



(b) Curva de ajuste de hiperparámetros del BRF.

Figura 5: Atributos más importantes y ajuste de hiperparámetros BRF en función de top-k features.

Para el caso de MLP aún efectuar el mismo procedimiento, no solo para features e hiperparametros, sino que también para encontrar mejor over y undersampling, dado que excepto para características (ya que se han utilizado todas las columnas, aspecto deficiente en la configuración de este modelo), las configuraciones fueron determinadas de manera euristica, aunque se determinaron los atributos de mayor importancia dado análisis de permutación.

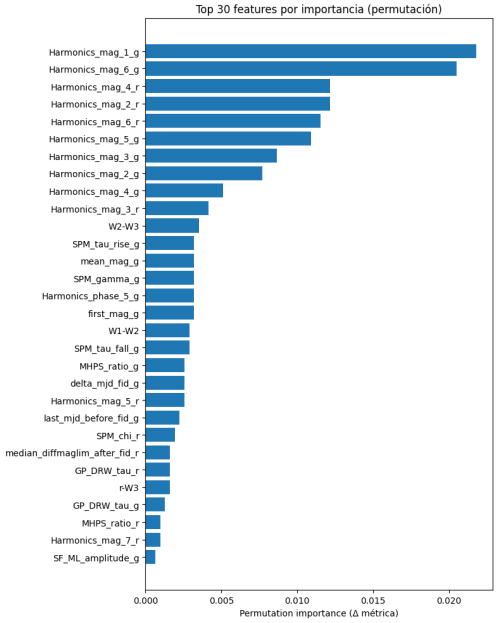


Figura 6: 30 atributos más importantes de MLP

7. Software

En el presente apartado serán brevemente discutidos aspectos técnicos de la programación del proyecto en Python.

7.1. Definición

Todo el código actualmente esta en un jupyter notebook, teniendo que ser ejecutado de manera secuencial para funcionar (aspecto a mejorar en la estructura del proyecto). Las 3 bases de datos son almacenadas en el Google Drive del usuario a ejecutar, específicamente en el directorio general `/content/drive/My Drive/*`.

7.2. Aspectos a Programar y Lenguaje de Programación

- **Lenguaje y entorno:** El proyecto se desarrolló en **Python** bajo **Google Colab**, integrando librerías de análisis de datos, visualización y aprendizaje automático. Se trabajó con tres fuentes principales de datos del catálogo ALeRCE en formato CSV y Parquet.
- **Manejo de datos:** Se utilizaron **pandas**, **numpy** y **pyarrow** para la lectura, manipulación y combinación de los conjuntos de datos. Las funciones más empleadas fueron `read_csv`, `read_parquet`, `merge` y `fillna`.
- **Visualización:** Se emplearon **matplotlib.pyplot** y **seaborn** para generar gráficos exploratorios y de resultados, tales como mapas de calor de correlación, gráficos de barras de importancia de atributos y proyecciones **UMAP**. Se configuraron colores y tamaños personalizados para representar las clases de supernovas.
- **Preprocesamiento y modelos clásicos:** El módulo **scikit-learn** (`sklearn`) se utilizó para dividir los datos (`train_test_split`), calcular métricas (`accuracy_score`, `precision_score`, `recall_score`, `f1_score`) e implementar los modelos **Random Forest** y **Balanced Random Forest**. Se empleó también `ConfusionMatrixDisplay` para graficar resultados.
- **Manejo de clases desbalanceadas:** Con la librería **imbalanced-learn** (`imblearn`) se aplicaron técnicas de muestreo como `RandomUnderSampler`, y `RandomOverSampler`, además del clasificador `BalancedRandomForestClassifier`, que balancea internamente las clases.
- **Red neuronal MLP:** Se implementó una red **Multilayer Perceptron (MLP)** en **PyTorch**, con arquitectura definida mediante la clase `Net`. Incluyó capas `Linear`, activaciones `Sigmoid` y `ReLU`, normalización por lotes (`BatchNorm1d`) y regularización con `Dropout`. Se entrenó con el optimizador `Adam` y la pérdida `CrossEntropyLoss` ponderada por clase.

- **Extensiones y mejoras del modelo:**
  - **MLPEncoder:** modelo con capa de embedding para extraer representaciones latentes.
  - **Center Loss:** penaliza la distancia intra-clase en el espacio latente, favoreciendo la compactación de embeddings.
  - **Focal Loss:** ajusta el peso de las clases difíciles mediante los parámetros α y γ, mejorando la robustez ante desbalance.
- **Optimización e importancia de atributos:** Se utilizó `RandomizedSearchCV` para la búsqueda de hiperparámetros del Balanced Random Forest, variando parámetros como profundidad, número de estimadores y fracción de atributos. Además, se calculó la **Permutation Importance** mediante una función implementada en PyTorch, evaluando la pérdida de desempeño al permutar cada atributo individualmente.
- **Reducción de dimensionalidad:** Se empleó **UMAP (Uniform Manifold Approximation and Projection)** para representar en 2D los embeddings de salida de los modelos y analizar visualmente la separabilidad entre clases.
- **Evaluación y métricas:** Se utilizaron métricas de **exactitud**, **exactitud balanceada**, **precisión**, **recuperación** y **F1-score** (macro y ponderado). También se aplicó **label smoothing** ($\epsilon = 0.02$) para reducir sobreajuste. Las matrices de confusión y las proyecciones UMAP se usaron para comparar el desempeño entre MLP, Random Forest y Balanced Random Forest.
- **Aspectos computacionales:**
  - Ejecución acelerada por GPU mediante `torch.device(cuda)` cuando disponible.
  - Reproducibilidad garantizada mediante semillas fijas (`random_state = 42`).
  - Estructura modular separando etapas de preprocesamiento, entrenamiento, evaluación y visualización.

8. Evaluación de Desempeño de Modelos

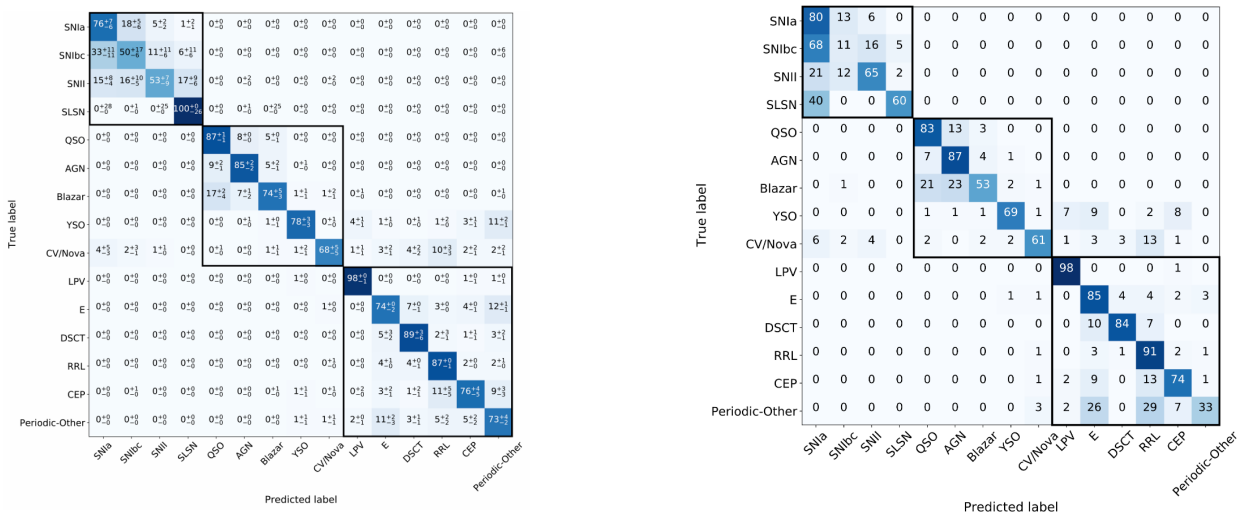
Para esta sección serán discutidos los aspectos relativos a métricas de desempeño utilizadas, formato de los resultados, tiempo de simulaciones, entre otros apartados importantes para posterior presentación de resultados preliminares.

8.1. Resultados Esperados

En lo relativo a resultados deseados, como objetivo de métricas y matriz de confusión se ocuparán los resultados descritos por ALeRCE para sus modelos bottom level con Balanced Random Forest y MLP.

Tabla 3: Tabla resumen de rendimiento de modelos bottom level descritos por ALeRCE.

Modelo	Precision	Recall	F1 Score
Balanced Radom Forest [Macro Average]	0.57 ± 0.01	0.76 ± 0.02	0.59 ± 0.01
MLP [Macro Average]	0.54	0.69	0.58



(a) Matriz de confusión BRF en bottom level

(b) Matriz de confusión MLP en bottom level

Figura 7: Matrices de confusión obtenidas por ALeRCE broker

8.2. Medidas de Desempeño de Modelos

Los desempeños comparables con los anteriores son las medidas Macro Averaged para Precision, Recall y F1 Score, las que toman sumatoria simple de cada una de las medidas para cada clase, para luego dividir este resultado por el total de clases. Por añadidura se agregan metricas de Accuracy y Balanced Accuracy, esta última sumando cada Accuracy de clase ponderada según la frecuencia de aparición de cada una.

8.3. Formato de Resultados

Se proporcionan matrices de confusión, visualización UMAP para conjunto de prueba (tanto matriz de entrada como de etiquetas predichas, no reales) y diversas métricas para todos los modelos.

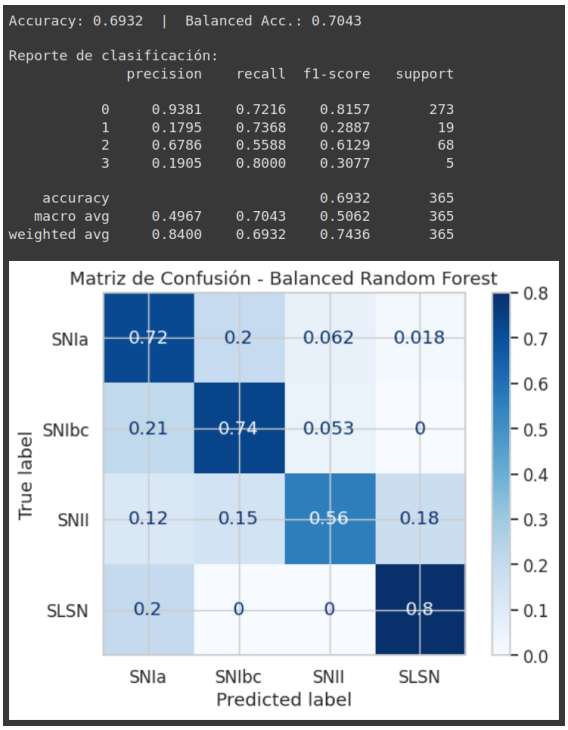
8.4. Número y Tiempo de Simulaciones

Los tiempos de entrenamiento de los modelos en sí son bastante rápidos dado el numero de estimadores y épocas reducidas, siendo aproximadamente de 6 segundos para MLP y de

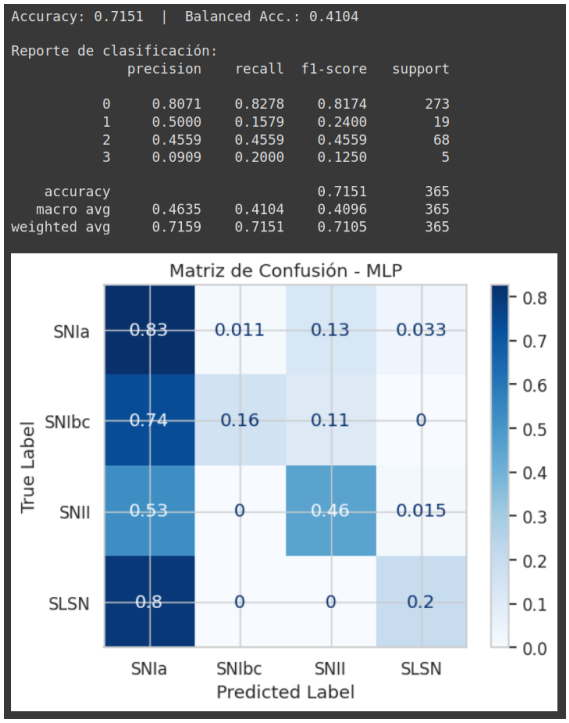
1 segundo para BRF, ayudándose MLP de la GPU T4 de colab para mejorar desempeño en funciones de Pytorch. Por otro lado, funciones aparte que usan UMAP y sklearn no se benefician de estos recursos, pero los tiempos siguen siendo reducidos. No obstante, el proceso de ajuste fino para BRF es el snippet que demora más en efectuarse, con un tiempo de 47 minutos en total, esto debido a la consideración de 25 combinaciones por cada reducción en número de características.

9. Resultados Preliminares

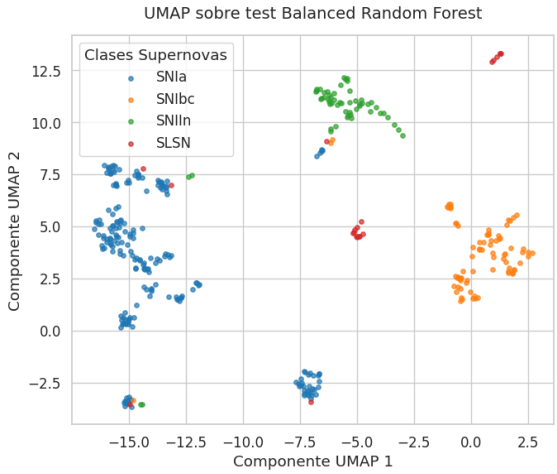
Las matrices, métricas y visualizaciones obtenidas son de un desempeño normal más no excelente en BRF, posiblemente debido al hyperparameter tuning en profundidad que se le hizo a este modelo; para el MLP los mismos resultados son muy deficientes, esto debido a que no se tuvo criterio de poda para atributos ruidosos, ni hubo evaluación de overfitting o ajuste fino de sus parámetros tanto en epocas, funciones de perdida, etc. En específico para UMAP, se puede apreciar como en BRF se tienen zonas de decisión rectangulares y con poco solapamiento de clases para casi todos los puntos, mientras que en MLP las 2 clases mayoritarias se agrupan en los centros asignados, con los targets SNIbc y SLSN presentando mucho acople con el target SNIa o directamente siendo outliers.



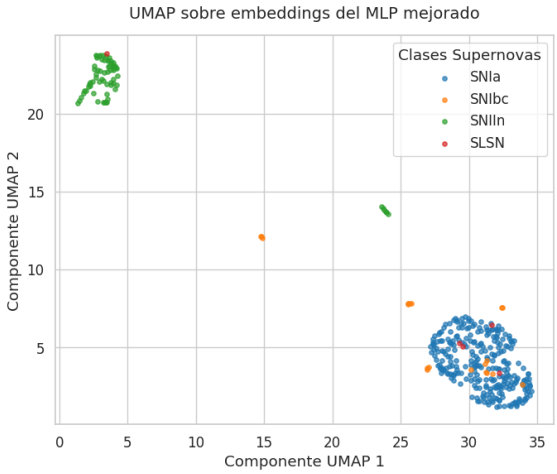
(a) Matriz de confusión y métricas BRF



(b) Matriz de confusión y métricas MLP



(c) Visualización UMAP BRF



(d) Visualización UMAP penúltima capa MLP

Figura 8: Desempeño y visualización del comportamiento de los modelos planteados

10. Próximos Pasos del Proyecto

- **Ajuste fino MLP:** Se efectuaran procedimientos similares a la optimización de BRF, no solo con hiperparametros sino que también con poda de características, además de ver curvas de entrenamiento y validación para determinar grado de sobreajuste.

- **Manipulación y agregación de características a lc_features:** Se evaluarán resultados al calcular nuevas características que ofrezcan una mayor diferenciación entre las clases de supernovas, de manera de mejorar desempeño y diferenciación en los MLP y BRF.
- **Implementación de t-SNE:** Se realizará nuevamente el experimento de Balanced Random Forest con reducción de dimensionalidad en los datos utilizando t-SNE. El objetivo es realizar una comparación de rendimiento entre t-SNE y UMAP utilizando criterios de representación, tiempo de procesamiento y variación de hiperparámetros relevantes.
- **Identificación de Outliers:** Implementar técnicas de clustering como DBScan para detectar outliers en los datos.
- **Implementación de Multiclass - SVDD:** Se implementará la red MLP junto a Multiclass - SVDD para clasificar y detectar anomalías, realizando una evaluación comparativa de rendimiento y eficiente entre la red MLP con y sin Multiclass - SVDD.

11. Referencias

- Sánchez-Sáez, P., Reyes, I., Valenzuela, C., Förster, F., Eyheramendy, S., Elorrieta, F., Bauer, F. E., Cabrera-Vives, G., Estévez, P. A., Catelan, M., Pignata, G., Huijse, P., De Cicco, D., Arévalo, P., Carrasco-Davis, R., Abril, J., Kurtev, R., Borissova, J., Arredondo, J., . . . Camacho-Iñiguez, E. (2021). Alert Classification for the ALeRCE Broker System: The Light Curve Classifier. The Astronomical Journal, 161(3), 141. https://arxiv.org/abs/2008.03311.

- Förster, F., Cabrera-Vives, G., Castillo-Navarrete, E., Estévez, P. A., Sánchez-Sáez, P., Arredondo, J., Bauer, F. E., Carrasco-Davis, R., Catelan, M., Elorrieta, F., Eyheramendy, S., Huijse, P., Pignata, G., Reyes, E., Reyes, I., Rodríguez-Mancini, D., Ruz-Mieres, D., Valenzuela, C., Álvarez-Maldonado, I., . . . Vergara, J. R. (2021). The Automatic Learning for the Rapid Classification of Events (ALeRCE) Alert Broker. The Astronomical Journal, 161(5), 242. https://arxiv.org/abs/2008.03303.

- Wang, H. (2023). A generalized scattering theory in quantum mechanics. Journal Of Physics Communications, 7(7), 075001. https://arxiv.org/abs/2307.01235.

- McInnes, L., Healy, J., & Melville, J. (2018, 9 febrero). UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction. arXiv.org. https://arxiv.org/abs/1802.03426.

- Van Der Maaten, L., & Hinton, G. (2008). Visualizing Data using t-SNE. https://www.jmlr.org/papers/v9/vandermaaten08a.html

- Y. Zhou, X. Liang, W. Zhang, L. Zhang, and X. Song, "VAE-based Deep SVDD for anomaly detection," Neurocomputing, vol. 453, pp. 131–140, Sep. 2021. https://doi.org/10.1016/j.neucom.2021.04.089