14 DE NOVIEMBRE DE 2025

AstronomIA

TRABAJO FINAL DE MINERÍA DE DATOS

REALIZADO POR: IVÁN TANG ZHU

Tabla de contenido

1. ENTENDIMIENTO DEL NEGOCIO

1.1. DESCRIPCIÓN DEL NEGOCIO

Ilustración 1 sacado de: https://ccapp.osu.edu/research/experiments-and-surveys/sdss

El Sloan Digital Sky Survey (SDSS) es un proyecto científico internacional dedicado a mapear de manera sistemática el cielo y registrar información detallada de millones de objetos astronómicos. Su negocio, entendido como su actividad principal, consiste en observar, medir, clasificar y poner a disposición pública datos astronómicos de alta calidad, utilizando tecnología avanzada de captura y análisis.

El SDSS opera desde el año 2000 y se basa en un telescopio óptico ubicado en el Observatorio Apache Point (Nuevo México, EE. UU.). Este telescopio está equipado con cámaras y espectrógrafos que permiten obtener imágenes profundas del cielo y medir el espectro de luz emitida por estrellas, galaxias y otros objetos.

El SDSS publica periódicamente sus datos en Data Releases (DR), cada una con nuevas observaciones y mejoras. La versión DR17, utilizada en este trabajo, contiene información actualizada de distintos objetos espaciales y representa una de las mayores bases de datos astronómicas disponibles

La base de datos sacado de Kaggle consta de 100.000 observaciones espaciales realizadas por el SDSSS (Sloan Digital Sky Survey DR17) es la publicación 17. Tiene 17 columnas de características que describen propiedades fotométricas y espectroscópicas de los objetos. Además, tiene una columna clase que la identifica como estrella, galaxia o cuásar.

Enlace a los datos:
https://www.kaggle.com/datasets/fedesoriano/stellar-classification-dataset-sdss17

1.2. DESCRIPCIÓN DEL PROBLEMA

En astronomía, la clasificación estelar consiste en identificar y categorizar los objetos celestes según sus características espectrales y fotométricas. La correcta diferenciación entre estrellas, galaxias y cuásares es una tarea fundamental para la comprensión del universo, ya que permite organizar catálogos astronómicos, estudiar la distribución de los objetos en el cielo y analizar su evolución.

Sin embargo, debido al enorme volumen de datos generados por proyectos de observación como el SDSS, el proceso manual de clasificación se vuelve cada vez más complejo, lento y propenso a errores. La acumulación de cientos de miles de observaciones con múltiples características dificulta que los astrónomos clasifiquen los objetos de manera eficiente solo mediante métodos tradicionales.

1.3. OBJETIVOS DE LA MINERÍA

- Desarrollar un modelo predictivo tipo **clasificación** para catalogar estrellas, galaxias y cuásares mediante el entrenamiento de diferentes modelos de Machine Learning.
- Optimizar los 3 **mejores modelos** para obtener el mejor modelo mediante uso de hiperparametrización con GridSearch y algoritmos genéticos.
- Realizar el **despliegue del mejor modelo** mediante interfaz gráfica con Streamlit lo que permite predecir el valor desde un formulario

1.4. DISEÑO DE SOLUCIÓN

Problema	Tipo de Minería	Tipo de aprendizaje	Requerimiento datos	Métodos	Evaluación
Clasificación estelar	Predictivo	Clasificación	-Histórico de datos -Variable objetivo -Relación de predictoras con objetivo	Árboles, Knn, Redes neuronales, Random Forest, métodos de ensamble (XGBoost, Voting Hard, Bagging)	Exactitud, precision, Recall, Area ROC, Matriz de confusión, F1 Score

- Evaluación esperada F1 Score > 80%

1.5. RECURSOS PARA CREACIÓN DEL MODELO Y PARA DESPLIEGUE

- HW:
  - Computador para el entrenamiento y optimización de modelos (en este caso de va a utilizar Google Colab)
  - CPU o un procesador de mayor nivel
  - Computador/servidor con el modelo desplegado (Se utiliza streamlit)
- SW:
  - Google Colab o entorno para ejecutar código Python
  - Python 3.8+
  - Librerías de Python: Pandas, Numpy, Matplotlib, Scikit-Learn, XGBoost, Pipes, streamlit.

2. ENTENDIMIENTO DE LOS DATOS

1.1. CICLO DE LOS DATOS

- Generación: El SDSS DR17 recopila información mediante observaciones en diferentes bandas fotométricas junto con espectros, posiciones y corregimiento al rojo.
- Almacenamiento: Base de datos
- Modificación: Limpieza de registros incompletos o con errores y posible reducción de variables
- Periodicidad: Genera datos continuamente a medida que se hacen nuevas observaciones, pero se publican en lotes (DR), el último DR19 fue en 2025, el que se maneja actualmente para el proyecto es el DR17 del 2021, por lo que cada año más o menos hay nuevos datos

1.2. DICCIONARIO DE DATOS

Variable	Descripción	Tipo
obj_ID	Identificador único del objeto.	ID numérico
alpha	Ascensión recta (Right Ascension) — posición en el cielo.	Numérico float
delta	Declinación — posición en el cielo.	Numérico float
u, g, r, i, z	Magnitudes en diferentes filtros del espectro electromagnético (ultravioleta a infrarrojo cercano).	Numérico float
run_ID, rerun_ID, cam_col, field_ID	Identificadores de la corrida, versión, columna de cámara y campo de observación.	ID numéricos
spec_obj_ID	Identificador del objeto en el catálogo espectroscópico.	ID numérico
class	Tipo de objeto celeste: GALAXY, STAR, o QSO	Categórico
redshift	Desplazamiento al rojo — indica la distancia y velocidad de alejamiento del objeto.	Numérico float
plate, MJD, fiber_ID	Parámetros técnicos relacionados con la observación espectral.	Numéricos

1.3. REGLAS DE CALIDAD

Variable	Regla calidad (valores válidos)
obj_ID	Valor único
alpha	0° a 360°
delta	-90 a 90°
u, g, r, i, z	9 a 30
redshift	-0.1 a 7
Cam_col	1 a 6
run_ID, rerun_ID, field_ID, spec_obj_ID, fiber_ID	ID numéricos
class	GALAXY, STAR QSO

3. PREPARACIÓN DE DATOS

3.1. INTEGRACIÓN

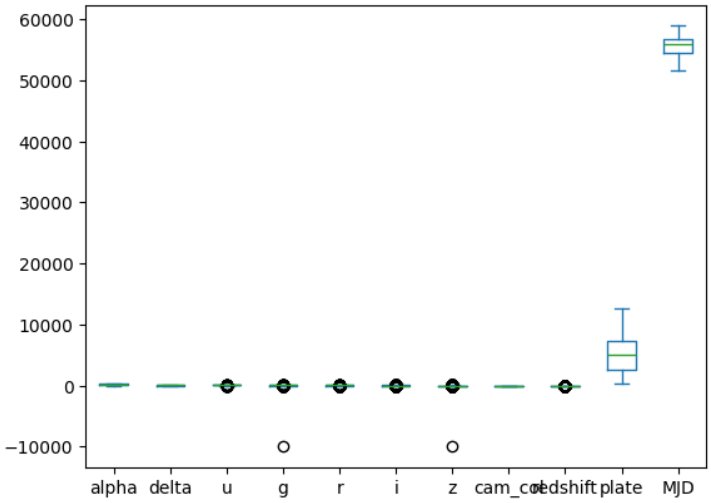
- No aplica al ya estar integrado en una tabla con todas las variables.

3.1. SELECCIÓN DE VARIABLES

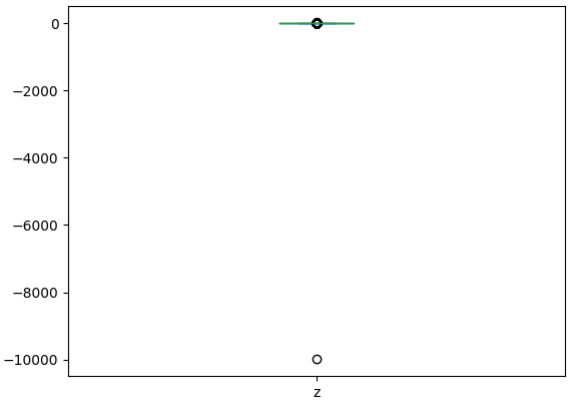
- Se eliminan variables ID: obj_ID, run_ID, rerun_ID, field_ID, fiber_ID. Ya que no aportan para el ejercicio de Machine Learning.

3.2. ESTADÍSTICA DESCRIPTIVA

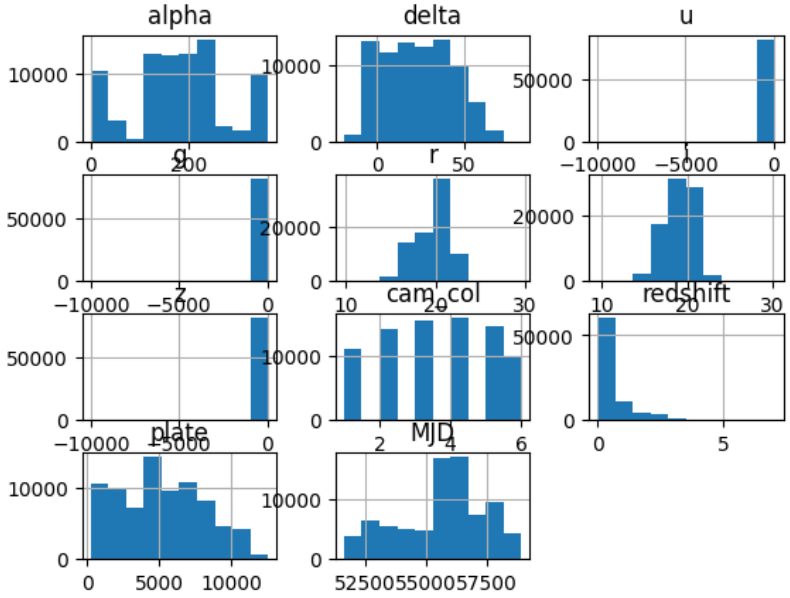
- Variables numéricas:
    - Desde el la gráfica box se puede ver que las variables tienen rangos totalmente diferentes y se sospecha de unos valores de ser atípicos



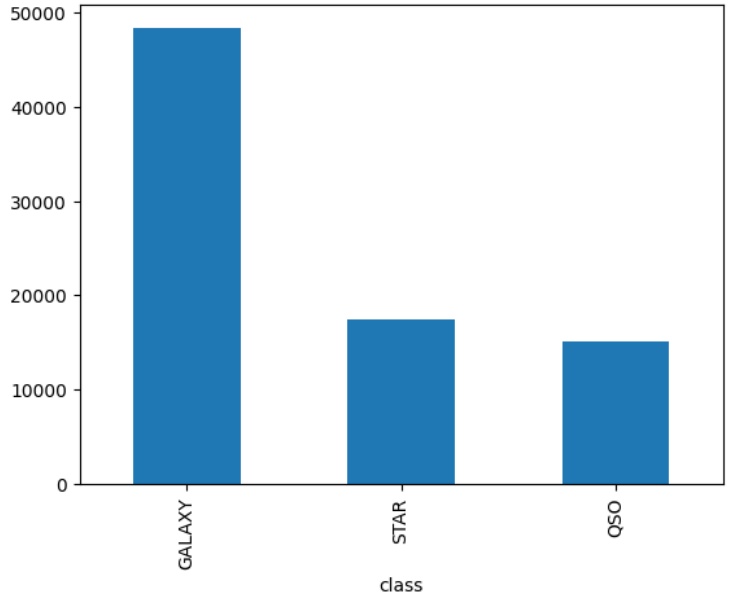
    - Viendo una de esas variables desde más cerca, definitivamente un valor de -10 000 es algo muy raro:



- Histogramas de las variables numéricas



- Variables categóricas (sólo aplica para variable objetivo):



-

3.3. LIMPIEZA DE ATÍPICOS

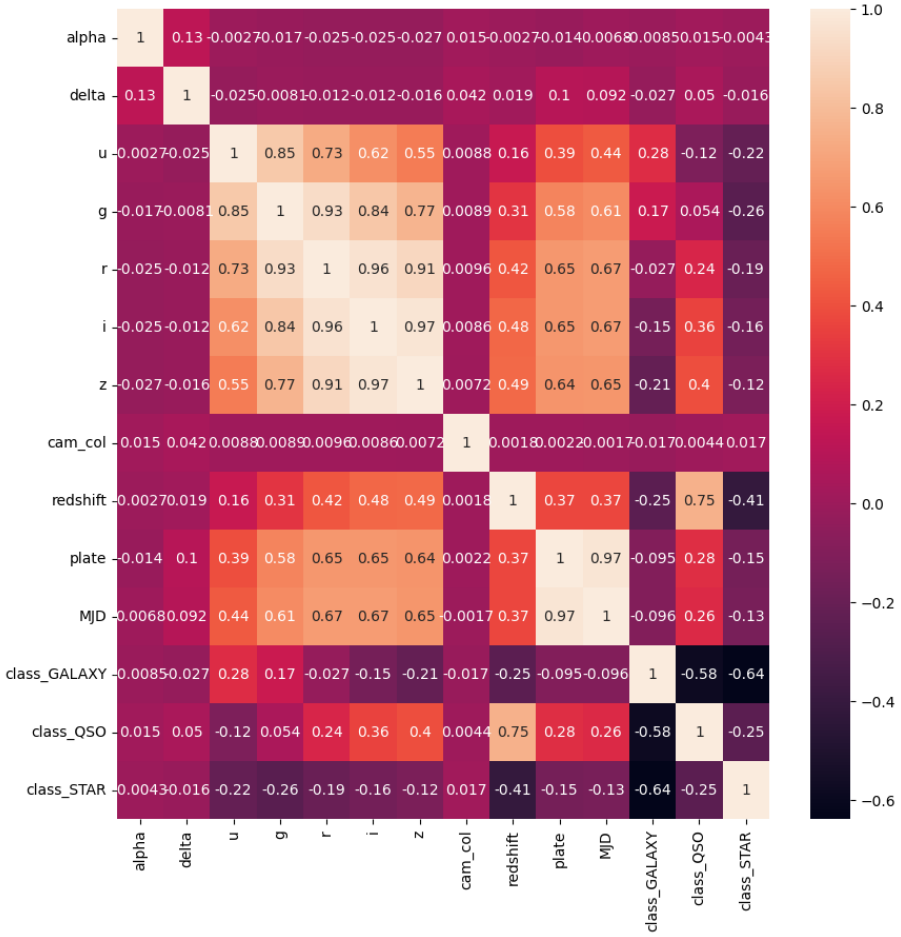
- En las variables: u, g, z se encontraron valores negativos, cosa que no es posible, es decir están fuera del rango.

3.4. LIMPIEZA DE NULOS

- Los únicos nulos que se encontraron fueron de los atípicos. Se imputa por la media al ser valores numéricos

3.5. ANÁLISIS DE CORRELACIONES PARA REDUNDANCIA

- Matriz de correlaciones:



- Variables u, g, r, i, z no se eliminan, aunque tengan valores muy altos entre ellos. Esto es porque son las variables más importantes, además de que tienen alta correlación es porque todas están en el mismo rango lo que puede verse como variables redundantes, pero en realidad cada una significa algo diferente por ejemplo u es ultravioleta y g es
- Pero si se eliminan otras variables como plate y MJD que tienen altas correlaciones, además de que es información que no trata sobre los objetos estelares.

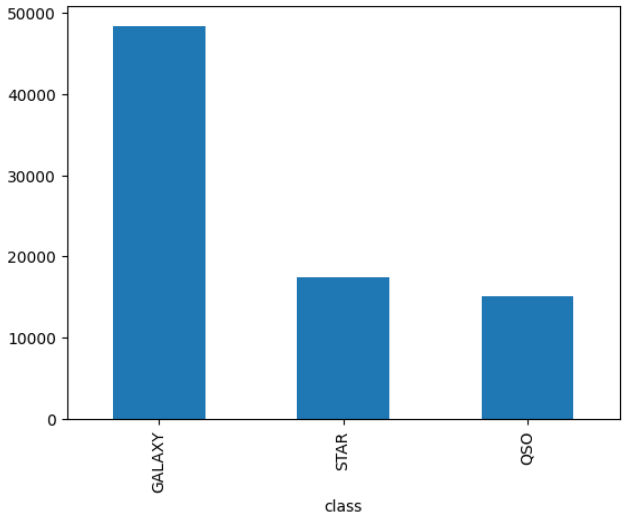
3.6. ANÁLISIS DE CORRELACIONES PARA IRRELEVANCIA (PREDICCIONES)

	class_STAR
alpha	-0.004331
delta	-0.015693
u	-0.215900
g	-0.259833
r	-0.193688
i	-0.156914
z	-0.123731
cam_col	0.016531
redshift	-0.414051
plate	-0.150517
MJD	-0.134331
class_GALAXY	-0.638615
class_QSO	-0.251025

- Se elimina alpha, delta y cam_col. Las dos primeras son ubicaciones estelares y la segunda es una propiedad del telescopio.

3.7. BALANCEO (CLASIFICACIÓN)0

- Se decide no realizar balanceo ya que las clases no están tan desbalanceadas



3.8. INGENIERÍA DE CARACTERÍSTICAS

3.8.1. CREACIÓN DE NUEVAS VARIABLES

- No se crearon nuevas variables

3.8.2. REDUCCIÓN DE DIMENSIÓN (OPCIONAL EN PREDICCIONES)

- No se aplicó Reducción de dimensionalidad principalmente porque con la limpieza de variables, al final no quedan muchas variables y como todas las variables predictoras son numéricas, no hay necesidad de hacer dummies (esto es el paso que aumenta las dimensiones)

3.8.3. TRANSFORMACIONES

- En métodos basado en arboles (Árbol de decisión, Random Forest y XGBoost): Discretizar
- En métodos numéricos (Knn, Red neuronal): Dummies a variables categóricas (en este caso no aplica), Codificar variable objetivo y normalizar datos numéricos

4. MODELAMIENTO, EVALUACIÓN E INTERPRETACIÓN

Nota: Se inicia haciendo separación 70-30 de los datos, de los cuales el 70% se aplica Cross Validation (Validación cruzada).

4.1. CONFIGURACIÓN MÉTODOS DE MACHINE LEARNING

- Configuración de la validación cruzada:

```
#Medidas de evaluación
scoring=('f1_macro', 'accuracy','precision_macro', 'recall_macro')

#Muestreo lineal
cv=StratifiedKFold(n_splits=10, shuffle=True, random_state=42)
```

```
scores = cross_validate(model_tree, X_train, Y_train, cv=cv, scoring=scoring, return_train_score=True, return_estimator=False)
```

  - Árbol:

```
#Arbol
from sklearn import tree
model_tree = tree.DecisionTreeClassifier(criterion='gini', min_samples_leaf=20, max_depth=None)
```

  - Knn:

```
#Método Perezoso
from sklearn.neighbors  import KNeighborsClassifier
model_knn = KNeighborsClassifier(n_neighbors=2, metric='euclidean')
```

  - Red Neuronal:

```
#Red neuronal
from sklearn.neural_network import MLPClassifier
model_rn = MLPClassifier(activation="relu",hidden_layer_sizes=(10,), learning_rate='adaptive',
                    learning_rate_init=0.02, momentum= 0.2, max_iter=500, verbose=False)
```

  - Random Forest:

```
#Random Forest
from sklearn.ensemble import RandomForestClassifier
model_rf= RandomForestClassifier(n_estimators=120,  max_samples=0.8,
                                 criterion='gini', max_depth=None, min_samples_leaf=20)
```

  - XGBoost:

```
#XGBoost
import xgboost as xgb
model_xgb = xgb.XGBClassifier(
    max_depth=20, learning_rate=0.1, n_estimators=100, subsample=0.8)
```

- Voting Hard:

```
#Votación hard: todos los modelos tiene el mismo peso
from sklearn.ensemble import VotingClassifier
clasificadores= [('knn', model_knn), ('net', model_rn), ('rf', model_rf)]
model_vot_hard = VotingClassifier(estimators=clasificadores, voting='hard')
```

- Bagging

```
#Bagging: Knn
from sklearn.ensemble import BaggingClassifier
modelo_base=KNeighborsClassifier(n_neighbors=1, metric='euclidean')

model_bag = BaggingClassifier(modelo_base, n_estimators=100, max_samples=0.7)
```

- Configuración de la Optimización por GridSearch (para este punto se hizo comparación de los 3 mejores y son: Árbol, Random Forest y XGBoost):
  - Árbol y mejores parámetros:

```
#  Arbol
from sklearn.tree import DecisionTreeClassifier
model_tree = DecisionTreeClassifier()

# Definir los hiperparametros
criterion=['entropy','gini'] #Indice de información
min_samples_leaf=[2,10,50,100,200,300] # Cantidad de registros por hoja
max_depth=[None, 10,20,50] #Niveles de profundidad
```

```
{'criterion': 'entropy', 'max_depth': None, 'min_samples_leaf': 50}
```

  - Random Forest y mejor modelo:

```
# Random Forest
from sklearn.ensemble import RandomForestClassifier

model_rf = RandomForestClassifier()

# Hiperparámetros para buscar
param_grid = {
    'n_estimators': [100, 200],
    'max_depth': [None, 20, 50],
    'min_samples_leaf': [10,50],
    'max_samples':[0.7, 0.8],
}
```

```
{'max_depth': None, 'max_samples': 0.8, 'min_samples_leaf': 10, 'n_estimators': 100}
```

- XGBoost y mejor modelo:

```
#  XGBoost
import xgboost as xgb
model_xgb = xgb.XGBClassifier()

# Definir los hiperparametros
learning_rate=[0.1,0.2] # Corrección en cada iteración
subsample=[0.8, 0.9] # Tamaño de subconjuntos
n_estimators=[50,100,200] # Cantidad arboles
max_depth=[None, 10,50,100] #Niveles de profundidad
```

```
{'learning_rate': 0.1, 'max_depth': 10, 'n_estimators': 100, 'subsample': 0.9}
```

- Optimización por algoritmos genéticos:
  - Nota: Dado que se demoraba mucho se decidió dejar la validación cruzada en 3. Además, iteraciones (generations) es 35 a excepción de Random Forest que fue de 10 por su demora

```
evolved_estimator = GASearchCV(
    estimator=model_rf,
    cv=3, #10
    scoring='f1_macro',
    population_size=20, #tamaño de la población es constante
    generations=10, #generaciones o iteraciones
    elitism=True, #selección de padres
    crossover_probability=0.4, #porcentaje para hacer recombinación
    mutation_probability=0.6, #porcentaje de mutación
    param_grid=param_grid,
    criteria='max',
    verbose=True)
```

  - Árbol y mejor modelo:

```
#  Arbol
from sklearn.tree import DecisionTreeClassifier
model_tree = DecisionTreeClassifier()

# Definir los hiperparámetros
param_grid = {
    'criterion': Categorical(['entropy', 'gini']),
    'max_depth': Integer(20, 200),
    'min_samples_leaf': Integer(2, 100)
}
```

```
{'criterion': 'entropy', 'max_depth': 63, 'min_samples_leaf': 19}
```

- Random Forest y mejor modelo:

```
#  Arbol
from sklearn.ensemble import RandomForestClassifier
model_rf = RandomForestClassifier()

# Definir los hiperparametros
param_grid = {
    'n_estimators': Integer(50,150),
    'criterion': Categorical(['entropy', 'gini']),
    'min_samples_leaf': Integer(2, 100),
    'max_samples': Continuous(0.8, 0.9)
}
```

```
{'n_estimators': 118, 'criterion': 'entropy', 'min_samples_leaf': 3, 'max_samples': np.float64(0.8754218028239278)}
```

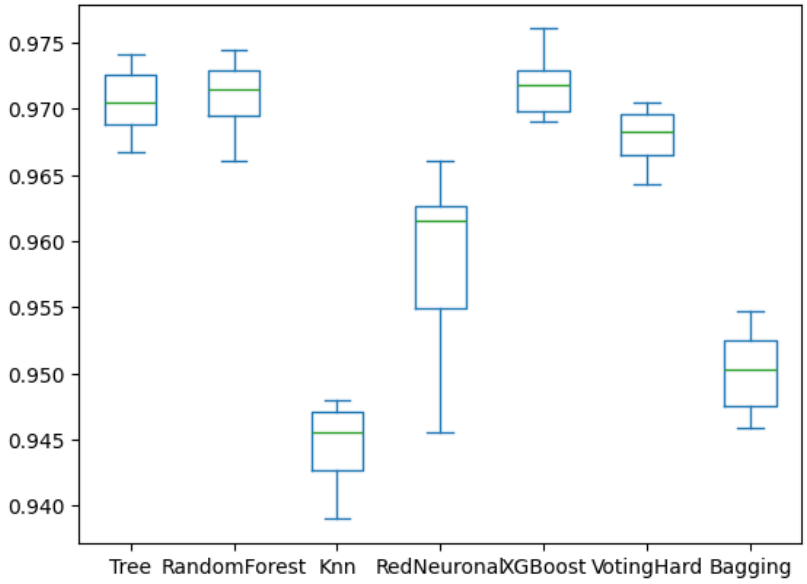
- XGBoost y mejor modelo:

```
#  XGBoost
import xgboost as xgb
model_xgb = xgb.XGBClassifier()

# Definir los hiperparámetros
param_grid = {
    'n_estimators': Integer(50,200),
    'learning_rate': Continuous(0.1, 0.5),
    'subsample': Continuous(0.7, 0.9),
}
```

```
{'n_estimators': 194, 'learning_rate': np.float64(0.1838653067795768), 'subsample': np.float64(0.8766591839052816)}
```

4.2. ANALISIS DE MEDIDAS DE CALIDAD

- Comparando la validación cruzada:
- 

    - Todos tuvieron un muy buen resultado, por arriba de 0,9 de F1 Score (media armónica). Para la siguiente fase de optimización se escogió los 3 mejores modelos: Árbol, RandomForest y XGBoost
- Comparando la optimización por GridSearch:

	GridSearch_Tree	GridSearch_XGBoost	GridSearch_RandomForest
f1 de la CV	0.969591	0.97238	0.972289

    - Tuvieron buenos resultados, pero no aumentó mucho
- Comparando la optimización por algoritmos genéticos:
    - Árbol: F1 Score de 0,970102

	Genetics_Tree
f1 de la CV	0.970102

    - Random Forest: F1 Score de

	Genetics_RandomForest
f1 de la CV	0.972671

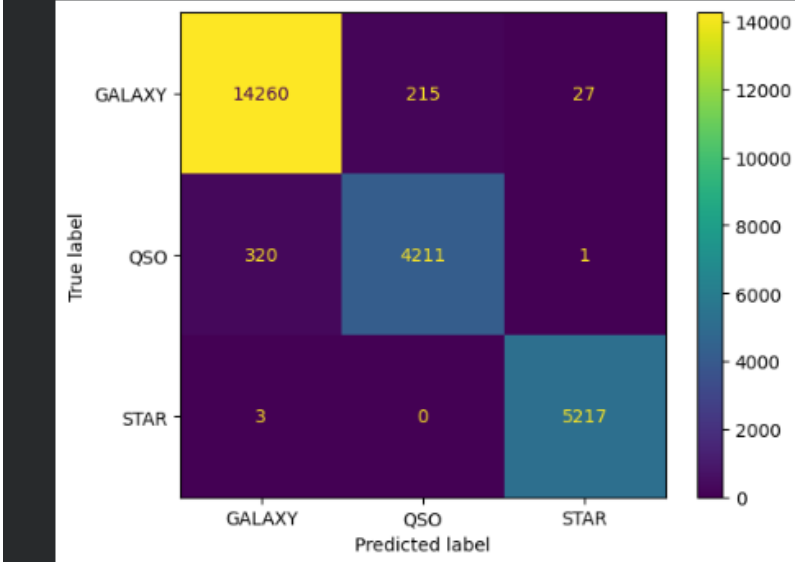
- XGBoost: F1 Score de 0,969167

	Genetics_XGBoost
f1 de la CV	0.969167

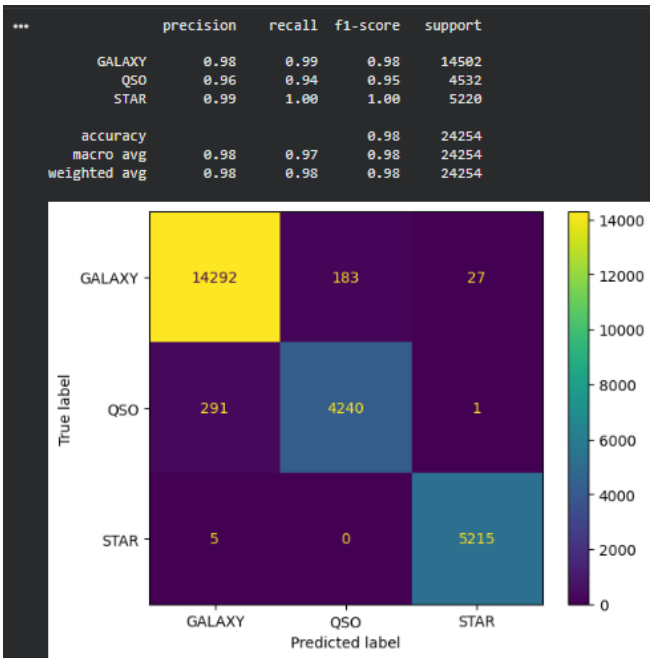
- En general tuvieron buenos resultados, pero comparando con con GridSearch realmente no hay mucho margen de mejora, después de todo con los primeros modelos que se realizaron ya están al 96%, por lo que con este proceso sólo podría mejorar entre 1% a 3%.

- Además, se muestra la evaluación con el 30% de los datos de los mejores modelos:
  - Árbol:

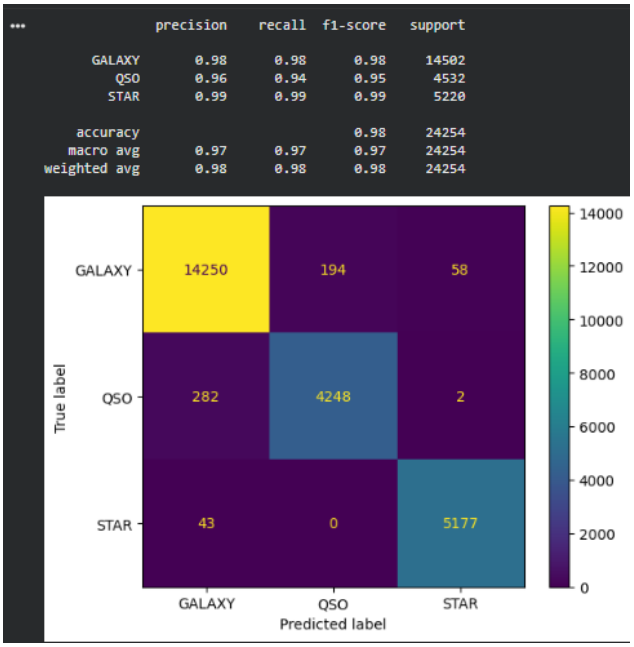
	precision	recall	f1-score	support
GALAXY	0.98	0.98	0.98	14502
QSO	0.95	0.93	0.94	4532
STAR	0.99	1.00	1.00	5220
accuracy			0.98	24254
macro avg	0.97	0.97	0.97	24254
weighted avg	0.98	0.98	0.98	24254



- Random Forest:



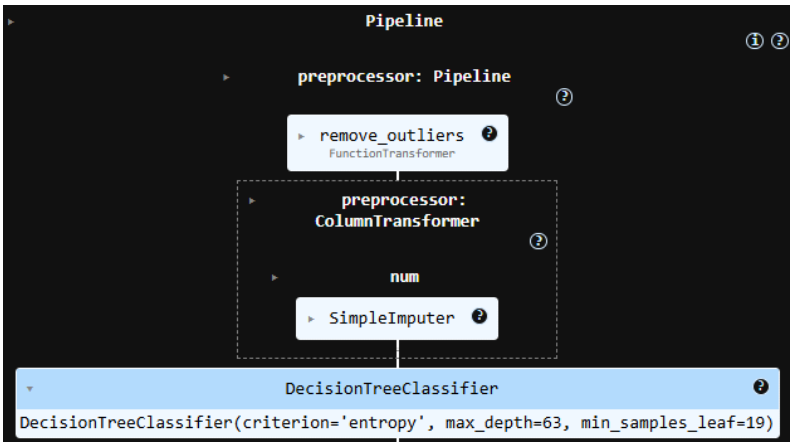
- XGBoost:



- Estas evaluaciones con el 30% en general fueron bien, con métricas similares a los de la validación cruzada que se hacen dentro de los algoritmos genéticos.

4.3. SELECCIÓN DEL MEJOR MODELO

- Al final se selecciona el Árbol de decisión, aunque no fue el mejor comparándolo a Random Forest o XGBoost, en realidad la diferencia es alrededor de 1%, un valor muy bajo, además de que, considerando la complejidad computacional, con un solo árbol da buenos resultados, no se necesitarían modelos como Random Forest o XGBoost que utilizan 100 o más árboles si con 1 sólo es suficiente.
- Ya teniendo seleccionado el modelo, ahora se entrena el modelo definitivo utilizando los mejores hiperparámetros hallados con los algoritmos genéticos. Además, se implementan Pipes
- El Pipe queda de la siguiente manera:



  - El Pipe es simple: Consta de una primera parte de eliminación de atípicos, una segunda parte de imputación de nulos y la tercera parte con el modelo entrenado utilizando los parámetros hallados con los algoritmos genéticos.
  - No hay procesamiento de variables categóricas porque no hay variables categóricas en las variables predictoras.

5. DESPLIEGUE

5.1. PREDICCIÓN DE DATOS FUTUROS

- Para este punto ya está el modelo guardado con el Pipe
- Primero se hace un despliegue utilizando dataset con datos futuros y se hace uso del pipe que contiene toda la preparación y modelo.
- Predicción del dataset:

	u	g	r	i	z	redshift	Predicción
0	23.87882	22.27530	20.39501	19.16573	18.79371	0.634794	GALAXY
1	20.11234	19.76321	19.32145	19.14200	19.03542	0.000012	STAR
2	21.54320	20.85432	20.11289	19.73102	19.50487	0.423568	QSO
3	22.78456	21.98765	21.10342	20.45231	20.23107	1.832451	QSO
4	19.45328	19.14273	18.93742	18.81231	18.76450	1.434147	QSO

-
- Ahora viendo que efectivamente funciona, se hace el despliegue el Streamlit, indicando los campos con sus respectivos rangos. Al final la interfaz visual queda:
    - El enlace es: https://astronom-ia.streamlit.app/

5.2. MONITOREO

- Esta etapa no estaba prevista por lo que se dejan algunas indicaciones:
    - Pruebas del modelo utilizando nuevos datos de los cuales ya se sabe el resultado, esto permitirá ver si la predicción es correcta
    - Pruebas de robustez, evaluando el comportamiento del modelo al tener datos incompletos y ruido en los datos.
    - Verificación con expertos del tema

5.3. CRONOGRAMA DE MANTENIMIENTO/RE-ENTRENAMIENTO

- El cronograma sería el siguiente:
    - 3 meses de evaluación del modelo mediante diferentes pruebas para verificar que la predicción sea correcta
    - 1 mes de Reentrenamiento (más o menos cada año se publica un nuevo lote de datos):
        - Utilizando datos más recientes: DR18 o DR19 (Data Release) del SDSS
        - Creación de nuevos modelos ajustando hiperparámetros.
        - Comparación de desempeño de nuevos modelos
        - Documentación de nuevos modelos