PROYECTO FINAL

Arias Apaza Jhon Hendrick, Medrano Callisaya Ivar Pedro
Universidad Mayor de San Andrés, La Paz, Bolivia

RESUMEN

Este proyecto utiliza un dataset extraído del SDSS para entrenar una red neuronal con el objetivo de clasificar galaxias, estrellas y quasares. El preprocesamiento incluyó imputación de valores faltantes, codificación categórica y normalización. Se construyó un modelo en TensorFlow utilizando múltiples pliegues estratificados, optimizado con Adam, alcanzando una precisión promedio de 96.1%. Aunque el modelo mostró robustez, se identificaron confusiones mínimas entre clases contiguas, destacando oportunidades de mejora en la metodología.

También se realizará una investigación acerca del uso del PCA y si esto lograría mejorar el rendimiento de nuestro modelo.

Palabras clave: Galaxias, estrellas, quasares, clasificación, red neuronal, preprocesamiento, TensorFlow, precisión, recall, F1-score, optimizador Adam.

ABSTRACT

This project uses a dataset extracted from the SDSS to train a neural network with the objective of classifying galaxies, stars and quasars. Preprocessing included imputation of missing values, categorical coding, and normalization. A model was built in TensorFlow using multiple stratified folds, optimized with Adam, achieving an average accuracy of 96.1%. Although the model showed robustness, minimal confusions between adjacent classes were identified, highlighting opportunities for improvement in the methodology.

Keywords: Galaxies, stars, quasars, classification, neural network, preprocessing, TensorFlow, precision, recall, F1-score, Adam optimizer.

1. INTRODUCCIÓN

La clasificación de objetos astronómicos como galaxias, estrellas y quasares es crucial en el ámbito de la astronomía computacional. Este trabajo busca entrenar una red neuronal que, a partir de datos fotométricos y redshift proporcionados por el SDSS, pueda categorizar eficazmente estos tres tipos de objetos. Con un dataset que contiene 100,000 entradas y 17 características, este estudio aborda el desafío de preprocesar datos, seleccionar un modelo adecuado y analizar su rendimiento. Las métricas de evaluación empleadas, como precisión, recall y F1-score, permiten evaluar la capacidad del modelo para manejar un conjunto de datos potencialmente desbalanceado.

2. MATERIALES Y MÉTODOS

Descripción del Dataset:

El Dataset cuenta con un total de 100,000 datos extraídos del SDSS(Sloan Digital Sky Survey), todos los datos están descritos en 17 columnas y una columna que almacena las clase entre galaxias, estrellas y quasares, esta columna se identifica cómo ”y”, siendo nuestra variable objetivo para la predicción/clasificación del modelo.

Objetivo de investigación: Realizar un análisis de los datos para lograr entrenar una red neuronal que logre categorizar entre las clases galaxia, estrellas y quasares.

Proceso de Análisis

Preprocesamiento: Se realizaron los siguientes pasos:

- Imputación de Valores Faltantes
- Codificación de valores categóricas usando One Hot Encoding
- Escalado de los datos numéricos usando Normalización
- Eliminación de columnas innecesarias.

Se reemplazaron los valores extremos (-9999) con la mediana y se eliminaron valores fuera del rango para las columnas fotométricas y redshift. No se usó balanceo de datos ya que el dataset original no cuenta con tanto desbalance como para que los modelos tengan dificultades para aprender los patrones.

Se usó One Hot Encoding ya que se tienen pocas categorías y no se tiene un orden ni relación entre ellas, tomando las categorías como independientes hará que la red neuronal no tenga problemas en la clasificación.

Selección del Clasificador: Se construyó un modelo de red neuronal con TensorFlow, realizando múltiples pliegues estratificados (100 splits), optimizado mediante Adam y evaluado con matriz de confusión y reporte de clasificación.

Se aplica un modelo secuencial en el que las capas se apilan una tras otra, se tiene una base de 128 neuronas y función de activación ReLU, que se define matemáticamente como:

$$f(z) = \frac{e^{z_i}}{\sum_{j=i}^{N} e^{z_i}} \quad (1)$$

donde z es el vector de entradas a la capa de salida y N es el número de clases. Esta función convierte los valores de salida en probabilidades, asegurando que la suma de todas las salidas sea 1, lo cual es crucial para la clasificación multiclase.

Compilación del Modelo

Se compila el modelo especificando el optimizador y la función de pérdida. En este caso, se usa el optimizador Adam y la función de pérdida categorical_crossentropy.

Se utiliza Adam por su gran rendimiento en problemas de clasificación el cual es nuestro objetivo.

La matemática detrás de este algoritmo es el siguiente:

Cálculo del promedio ponderado exponencial de los gradientes

$$v_{dw} = \beta_1 v_{dw} + (1 - \beta_1)\nabla_\theta J(\theta) \quad (2)$$

$$v_{db} = \beta_1 v_{db} + (1 - \beta_1)\nabla_\theta J(\theta) \quad (3)$$

Explicación:

v_{dw} son los promedios ponderados exponencialmente de los gradientes para los pesos y el sesgo, respectivamente.

β_1 Es el coeficiente que determina el peso de los gradientes anteriores con un valor típico de 0.9.

Cálculo del promedio ponderado exponencial de los cuadrados de los gradientes

$$s_{dw} = \beta_2 s_{dw} + (1 - \beta_2)(\nabla\theta J(\theta))^2 \quad (4)$$

$$s_{db} = \beta_2 s_{db} + (1 - \beta_2)(\nabla\theta J(\theta))^2 \quad (5)$$

Explicación:

s_{dw} son los promedios ponderados exponencialmente de los cuadrados de los gradientes para los pesos y el sesgo, respectivamente.

β_2 es el coeficiente que determina el peso de los gradientes cuadrado anteriores, con un valor típico de 0.999.

Corrección del Sesgo

Adam aplica una corrección del sesgo dado que los promedios ponderados exponencialmente están sesgados a 0 en las primeras iteraciones.

$$\hat{v}dw = \frac{vdw}{1-\beta_1^t} \quad (6)$$

$$\hat{sd}w = \frac{vdw}{1-\beta_1^t} \quad (7)$$

Actualización de Parámetros

$$\theta = \theta - n\frac{\hat{v}dw}{\sqrt{\hat{sd}w + \epsilon}} \quad (8)$$

Explicación:

n es la tasa de aprendizaje.

ϵ es un término pequeño para evitar la división por cero, con un valor típico de 10^{-8}.

Uso de PCA (ANÁLISIS DE COMPONENTES PRINCIPALES)

Primero hagamos un análisis de la matemática detrás de este método.

Eigenvectores y eigenvalores

Los eigenvectores y eigenvalores corresponden a números y vectores asociados a matrices cuadradas. Dada una matriz A de n x n, su eigenvector $\overline{v}$ de una matriz n x 1 tal que:

$$A * \overline{V} = \lambda * \overline{V} \quad (9)$$

donde el número λ es el eigenvalor, un valor escalar real asociado con el eigenvector.

Siempre que sean compatibles en tamaño, podemos multiplicar dos matrices entre sí. Los **eigenvectores**, autovectores o vectores propios son un caso especial de esta operación entre una matriz y un vector, siendo los eigenvectores de una matriz todos aquellos que, al ser multiplicados por esta matriz, resulten en el mismo vector o en un múltiplo entero de él. Considerando el siguiente ejemplo

$$\begin{pmatrix} 2 & 3 \\ 2 & 1 \end{pmatrix} \times \begin{pmatrix} 6 \\ 4 \end{pmatrix} = \begin{pmatrix} 24 \\ 16 \end{pmatrix} = 4 \times \begin{pmatrix} 6 \\ 4 \end{pmatrix} \quad (10)$$

el vector resultante $\left(\frac{24}{16}\right)$ es múltiplo entero del vector original: 4 * $\left(\frac{6}{4}\right)$ lo que es igual a decir que el vector resultante es 4 veces el vector original. Es por tanto, un eigenvector de la matriz.
Entre las **propiedades** de los eigenvectores se encuentran:

• Solo las matrices cuadradas tienen eigenvectores, pero no todas las matrices cuadradas los tienen. Dada una matriz n×n con eigenvectores, el número existente de ellos es n.

• Un eigenvector escalado, es decir, si se multiplica por cierto valor antes de multiplicarlo por una matriz, el eigenvector continuará manteniendo su propiedad, ya que solo se cambia su longitud, no su dirección (es la dirección y no la longitud la que determina la propiedad de eigenvector de un determinado vector). Con respecto al ejemplo anterior, el vector se ha podido escalar de la siguiente forma:

$$2 \times \begin{pmatrix} 3 \\ 2 \end{pmatrix} = \begin{pmatrix} 6 \\ 4 \end{pmatrix} \tag{11}$$

Independientemente del número de dimensiones, todos los eigenvectores de una matriz son perpendiculares. Esto significa que podemos expresar los datos respecto a estos eigenvectores.

Es frecuente escalar los eigenvectores para que tengan una longitud de 1, de manera que todos tengan la misma longitud. Siguiendo con el ejemplo anterior,

$$\begin{pmatrix} 3 \\ 2 \end{pmatrix} \tag{12}$$

es un eigenvector, cuya longitud es

$$\sqrt{\left(3^2 + 2^2\right)} = \sqrt{13} \tag{13}$$

El escalado de este eigenvector se llevaría a cabo dividiéndolo entre esta cantidad:

$$\begin{pmatrix} 3 \\ 2 \end{pmatrix} \div \sqrt{13} = \begin{pmatrix} 3/\sqrt{13} \\ 2/\sqrt{13} \end{pmatrix} \tag{14}$$

Cada uno de los componentes principales generados por PCA se corresponde a un eigenvector (dirección).

Por otro lado, los **eigenvalores** o valores propios son los valores con los que se multiplica el eigenvector y que dan lugar al vector original. En el ejemplo anterior, el eigenvalor asociado al eigenvector se corresponde con el valor 4. Los eigenvalores miden la cantidad de variabilidad retenida por cada componente principal (siendo mayores para la primera componente principal que para el resto), por lo que pueden usarse para determinar el número de componentes principales a retener.

• Un eigenvalor > 1 indica que la componente principal explica más varianza de lo que lo hace una de las variables originales, estando los datos estandarizados.

RESULTADOS

Para tener un dataset limpio para realizar el entrenamiento de un modelo de clasificación es necesario que no se tengan datos vacíos que pueden entorpecer el resultado final, por lo que primero se debe realizar una búsqueda de datos faltantes y en este caso reemplazarlos por con la mediana encontrada de los datos totales.

```
obj_ID         0
alpha          0
delta          0
u              0
g              0
r              0
i              0
z              0
run_ID         0
rerun_ID       0
cam_col        0
field_ID       0
spec_obj_ID    0
class          0
redshift       0
plate          0
MJD            0
fiber_ID       0
dtype: int64
```

Figura 1. Identificación de valores faltantes
Cómo se logra ver en la figura 1 en el dataset no existían valores faltantes

Codificación de valores categóricas usando ONE HOT ENCODING

La columna objetivo “y”, que contiene las clases galaxia, estrellas y quasares que serán usadas para el entrenamiento serán codificadas para que el modelo logre entender las variables.

class_GALAXY	class_QSO	class_STAR
True	False	False
True	False	False
True	False	False
True	False	False
True	False	False

Figura 2. La columna class luego de realizar One Hot Encoding

Eliminación de columnas irrelevantes

Realizado el análisis se eliminaron las columnas que son innecesarias para la clasificación de las clases, las cuales son “'obj_ID', 'run_ID', 'rerun_ID', 'cam_col', 'field_ID', 'spec_obj_ID', 'plate', 'MJD', 'fiber_ID'”.

	alpha	delta	u	g	r	i	z	redshift	class_GALAXY	class_QSO	class_STAR
0	135.689107	32.494632	23.87882	22.27530	20.39501	19.16573	18.79371	0.634794	True	False	False
1	144.826101	31.274185	24.77759	22.83188	22.58444	21.16812	21.61427	0.779136	True	False	False
2	142.188790	35.582444	25.26307	22.66389	20.60976	19.34857	18.94827	0.644195	True	False	False
3	338.741038	-0.402828	22.13682	23.77656	21.61162	20.50454	19.25010	0.932346	True	False	False
4	345.282593	21.183866	19.43718	17.58028	16.49747	15.97711	15.54461	0.116123	True	False	False

Figura 3. Datos después de la eliminación de las columnas innecesarias.

Normalización de columnas numéricas

Después de identificar las columnas numéricas se usa MinMax Scaler para la normalización de los datos.

	alpha	delta	u	g	r	i	z	redshift	class_GALAXY	class_QSO	class_STAR
0	0.376905	0.503802	0.591347	0.558050	0.535344	0.427665	0.464377	0.090699	True	False	False
1	0.402286	0.491812	0.632603	0.584423	0.646203	0.515986	0.607035	0.111322	True	False	False
2	0.394960	0.534139	0.654888	0.576463	0.546218	0.435729	0.472194	0.092042	True	False	False
3	0.940947	0.180600	0.511384	0.629186	0.596946	0.486717	0.487460	0.133213	True	False	False
4	0.959118	0.392679	0.387463	0.335579	0.337999	0.287021	0.300043	0.016592	True	False	False

Figura 4. Columnas después de haber realizado la normalización.

Visualización de la distribución de las clases

Se muestra la distribución de clases en un conjunto de datos, con la clase GALAXY siendo la más representada, seguida por STAR y QSO, que tienen menos muestras. Esto indica un desbalance de clases, donde la clase GALAXY predomina significativamente sobre las otras.

```
Distribución de clases:
class
GALAXY    59445
STAR      21594
QSO       18961
Name: count, dtype: int64
```

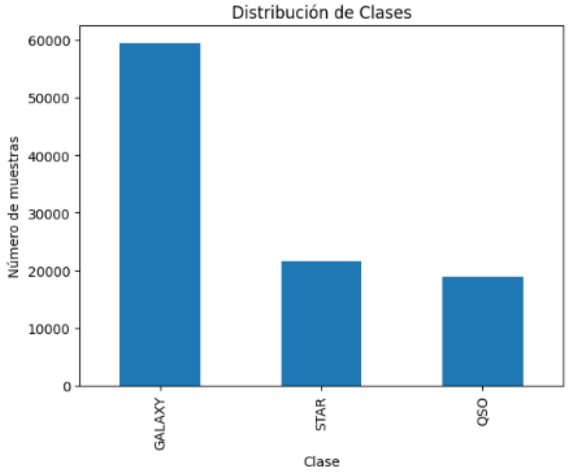

Figura 5. Representación por barras de la distribución de las clases.

Aplicación de la Red Neuronal

Luego de entender la matemática detrás de la implementación de nuestra red neuronal podemos usar librerías incluidas en python para hacer el código menos tedioso, el resultado es el siguiente. Se mostrarán datos del split 100

```
Matriz de Confusión en el pliegue 100:
[[9212  117   93]
 [ 145 3269    2]
 [ 221   15 2766]]
```

Figura 6. Matriz de confusión del modelo en el pliegue 100, mostrando las predicciones correctas e incorrectas para las clases galaxia, estrella y quasar.

```
Reporte de Clasificación:
              precision    recall  f1-score   support

           0       0.96      0.98      0.97      9422
           1       0.96      0.96      0.96      3416
           2       0.97      0.92      0.94      3002

    accuracy                           0.96     15840
   macro avg       0.96      0.95      0.96     15840
weighted avg       0.96      0.96      0.96     15840
```

Figura 7. "Reporte de métricas del modelo para el pliegue 100, incluyendo precisión, recall, F1-score y soporte de cada clase, junto con los promedios macro y ponderados."

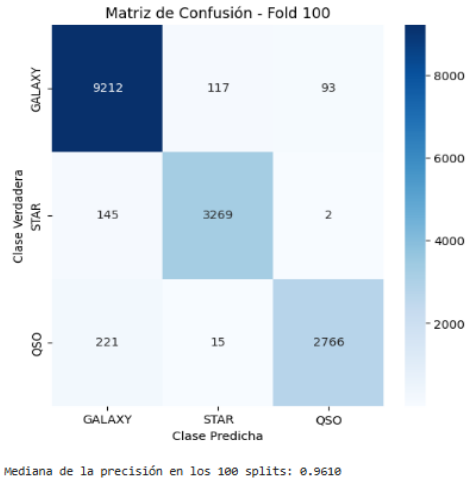

Figura 8. Visualización gráfica de la matriz de confusión del pliegue 100.

Rendimiento general del modelo:

El modelo mantiene un desempeño sólido con una

Análisis por clase:

- **GALAXY (clase 0):** Tiene la mayor cantidad de predicciones correctas, con un alto recall (0.98) y precisión (0.96), indicando que el modelo identifica esta clase de manera consistente.

- **STAR (clase 1):** Presenta un rendimiento equilibrado, aunque muestra una ligera disminución en la precisión (0.96) debido a confusiones con otras clases.
- **QSO (clase 2):** Tiene un desempeño sólido, con un F1-score de 0.94, aunque sufre leves confusiones con las clases GALAXY y STAR.

Matriz de confusión:
La gráfica de la matriz de confusión indica que la mayoría de las predicciones incorrectas ocurren en categorías contiguas (e.g., STAR confundido con GALAXY). Sin embargo, estas confusiones son mínimas en comparación con el total de instancias clasificadas.
Promedios macro y ponderados:
Los promedios macro (0.96) y ponderados (0.96) confirman un balance en el rendimiento entre las clases, lo que refuerza la eficacia general del modelo, incluso en un dataset con distribuciones posiblemente desbalanceadas.

4.DISCUSIÓN

Los resultados obtenidos reflejan un alto desempeño del modelo, con métricas consistentes en precisión, recall y F1-score para todas las clases. Sin embargo, se identificaron confusiones entre las clases STAR y GALAXY, lo que sugiere la necesidad de ajustar aún más la metodología para mejorar su precisión en estas categorías.

Entre las ventajas de la metodología empleada destacan su robustez frente a distribuciones desbalanceadas y su capacidad de generalización. Sin embargo, una desventaja importante es el tiempo requerido para entrenar el modelo, lo que podría limitar su uso en aplicaciones de tiempo real.

5. CONCLUSIONES

El modelo propuesto logró clasificar con éxito galaxias, estrellas y quasares con una precisión promedio del 96.1%, demostrando su efectividad para tareas de clasificación multiclase en astronomía. Si bien se observan confusiones menores entre las clases, el modelo mostró un desempeño equilibrado y una robustez frente a datos desbalanceados. Futuras mejoras podrían enfocarse en optimizar la eficiencia del tiempo de entrenamiento y en reducir las confusiones específicas entre clases. Este enfoque resalta la importancia de técnicas avanzadas de machine learning en la exploración del universo y la clasificación de datos astronómicos.

6. BIBLIOGRAFÍA

[1] F. Munini, D. Maggio, S. Bonometto, and M. Colavincenzo, “Dark Matter Halos in the Background Metric”, *Universe*, vol. 8, no. 2, p. 120, Feb. 2022. Disponible: https://www.mdpi.com/2218-1997/8/2/120

Recursos en línea

[2] F. Soriano. Stellar Classification Dataset, *Kaggle*, [Online]. Disponible: https://www.kaggle.com/datasets/fedesoriano/stellar-classification-dataset-sdss17?resource=download

[3] Codificando Bits. Función de Activación, [Online]. Disponible: https://codificandobits.com/blog/funcion-de-activacion/

[4] DataCamp. Adam Optimizer Tutorial, [Online]. Disponible: https://www.datacamp.com/es/tutorial/adam-optimizer-tuto