Instituto Tecnológico de Estudios Superiores de Monterrey

Inteligencia artificial avanzada a la ciencia de datos (TC3006C)

Semestre Agosto-Diciembre 2023

Módulo 2: Análisis y Reporte sobre el desempeño del modelo.

Alumno:

Pablo Gabriel Galeana Benítez A01735281

Profesor:

Mtro. Jesús Adrián Rodríguez Rocha

10 de septiembre de 2023

I. INTRODUCCIÓN

Los modelos de clasificación son esenciales en aprendizaje maquina para convertir datos en . El desarrollo eficaz de estos modelos implica enfrentar desafíos como el sobreajuste, subajuste, y entender la interacción entre sesgo y varianza. Además, el refinamiento de hiperparámetros es crucial para optimizar el rendimiento del modelo. Este trabajo se centra en estos conceptos clave, utilizando la clasificación estelar como caso de estudio.

A. Objetivo

Demostrar la competencia de comprensión y aplicación de conceptos clave en aprendizaje maquina, tales como sobreajuste, subajuste, sesgo, varianza en el entrenamiento, así como separación de datos y optimización de hiperparámetros, mediante la evaluación de un modelo.

B. Descripción

Se utilizará un conjunto de datos de clasificación proveniente de Kaggle para implementar el modelo KNN y posteriormente refinar los hiperparámetros, evaluando las métricas de exactitud y puntuación F1, así como el tiempo de computación requerido. Se utilizara una técnica de regularización y optimización de hiperparámetros como lo es búsqueda aleatoria para mejorar el rendimiento del modelo. Posteriormente, se aplicarán y evaluarán otros seis modelos, tanto simples (Arboles de decisión, Maquina de Soporte Vectorial y Regresión Logística) como de ensamble (Bosques Aleatorios, Potenciación de Gradiente y Clasificación por Votación), para comparar su rendimiento con el modelo KNN refinado.

II. DESARROLLO

A. ETL

En el contexto de este documento, ETL se referirá al proceso de exploración, transformación y limpieza. Este tiene por objetivo preparar los datos para la implementación de un modelo o varios.

1. Exploración

En primer lugar, la exploración de los datos, en especifico de los predictores, con el fin de conocer la clase de variables con las que se esta trabajando y de igual forma decidir cuales podrían ser eliminadas de forma inmediata. Se muestra en la tabla I.

Se observa que solo tenemos variables numéricas y que ninguna de las predictores contiene valore nulos. Haciendo un recuento simple se nota que sí existen datos atípicos e incluso extremos, sin embargo, vale la pena considerar que se tienen cien mil observaciones, lo que implica que aunque existan dichos datos, se cuenta con suficientes que no lo son para hacer un correcto entrenamiento. Idealmente, este análisis se debería de llevar acabo de forma visual y con métricas como el rango intercuartil como limite de los datos, pero debido al enfoque que se trata de dar y las conclusiones que se plantean dentro de la introducción, no se llevará acabo en esta ocasión.

2. Transformación

En segundo lugar, viene la transformación de los datos. Debido a que es un modelo de clasificación, la transformación más conveniente y que acelerará el proceso de computación y convergencia es la estandarización. La estandarización (o normalización de puntuaciones Z) tiene como resultado que las características se ajusten de manera que su media sea 0 y su desviación estándar sea 1. [1]

Se muestra en la tabla II la descripción de los datos ahora estandarizados. Se menciona y destaca que en la programación de los algoritmos se utilizará la transformación de forma diferente pero con el mismo resultado.

3. Limpieza

En tercer y ultimo lugar, la limpieza de los datos. En este caso, si se puede dar procedimiento, ya que existen dos predictores que no aportarían información al modelo. La primera de ellas es $rerun\_ID$, la cual si se observa, tiene todas sus métricas en 0, lo que implica que todos los registros tienen el mismo valor. Si bien durante el entrenamiento podrían establecerse algunas penalizaciones en determinados modelos que causen que predictores de este estilo tengan pesos nulos (Lasso en RL), sería más conveniente eliminarla desde el principio para ahorrar tiempo al disminuir el trabajo computacional. La segunda es obj_ID, la cual no es tan obvia, pero el nombre se refiere al identificador del objeto en caracterización, es decir, un mera convención o administración por lo que no tendría significancia para el modelo. Estas variables se deben de retirar para ayudar al modelo a tener un mejor desempeño.

B. Selección del modelo

Como paso intermedio entre el ETL y la implementación, es necesario seleccionar un modelo. Existen bastantes modelos de clasificación multiclase que trabajan bastante bien con variables continuas como predictores y hay varios criterios a considerar para definir cual es

Feature	Conteo	Media	Des. Est	Minimo	25 %	50 %	75 %	Maximo
obj_ID	100000	1.24E+18	8.4386E+12	1.24E+18	1.24E+18	1.24E+18	1.24E+18	1.24E+18
alpha	100000	177.629117	96.5022409	0.00552783	127.518222	180.9007	233.895005	359.99981
delta	100000	24.1353046	19.6446654	-18.7853281	5.14677086	23.6459223	39.9015501	83.0005186
u	100000	21.9804683	31.7692908	-9999	20.3523525	22.179135	23.68744	32.78139
g	100000	20.531387	31.7502923	-9999	18.96523	21.099835	22.1237675	31.60224
r	100000	19.6457621	1.85475969	9.82207	18.1358275	20.12529	21.044785	29.57186
i	100000	19.0848542	1.75789479	9.469903	17.732285	19.405145	20.396495	32.14147
z	100000	18.6688103	31.7281518	-9999	17.4606775	19.004595	19.92112	29.38374
run_ID	100000	4481.36606	1964.76459	109	3187	4188	5326	8162
rerun_ID	100000	301	0	301	301	301	301	301
cam_col	100000	3.51161	1.58691222	1	2	4	5	6
field_ID	100000	186.13052	149.011073	11	82	146	241	989
spec_obj_ID	100000	5.78E+18	3.32E+18	3.00E+17	2.84E+18	5.61E+18	8.33E+18	1.41E+19
redshift	100000	0.5766608	0.73070728	-0.00997067	0.05451684	0.42417325	0.70415433	7.011245
plate	100000	5137.00966	2952.30335	266	2526	4987	7400.25	12547
MJD	100000	55588.6475	1808.48423	51608	54234	55868.5	56777	58932
fiber_ID	100000	449.31274	272.498404	1	221	433	645	1000

Cuadro I: Descripción de los datos

Feature	Conteo	Media	Des. Est	Minimo	25 %	50 %	75 %	Maximo
obj_ID	100000	1.76E-11	1.000005	-2.22538024	-0.65895694	-0.14915779	0.42998921	1.87349746
alpha	100000	-3.97E-16	1.000005	-1.84062541	-0.51927442	0.0339018	0.58305558	1.88981731
delta	100000	1.17E-16	1.000005	-2.18486018	-0.96660484	-0.02491184	0.80257536	2.99651366
u	100000	2.28E-17	1.000005	-315.431359	-0.05124836	0.00625345	0.05373051	0.33998164
g	100000	2.82E-17	1.000005	-315.574464	-0.04932757	0.0179038	0.05015351	0.34868682
r	100000	4.59E-16	1.000005	-5.29650353	-0.81409047	0.25854037	0.7542917	5.35171619
i	100000	-2.33E-15	1.000005	-5.46961022	-0.76942943	0.18220224	0.74614666	7.42745307
z	100000	-4.65E-17	1.000005	-315.735974	-0.03807782	0.01058323	0.03947018	0.33771218
run_ID	100000	-1.43E-16	1.000005	-2.22540041	-0.65879268	-0.14931434	0.42989281	1.87332994
rerun_ID	100000	0	0	0	0	0	0	0
cam_col	100000	-6.76E-17	1.000005	-1.58271045	-0.95255272	0.30776273	0.93792046	1.56807819
field_ID	100000	7.33E-17	1.000005	-1.17529115	-0.69881411	-0.26931368	0.36822602	5.38801231
spec_obj_ID	100000	2.13E-16	1.000005	-1.64992899	-0.88439985	-0.05084211	0.7666252	2.50994568
redshift	100000	-3.99E-17	1.000005	-0.80283093	-0.71457694	-0.20868591	0.17448048	8.80601108
plate	100000	9.64E-17	1.000005	-1.64990973	-0.88440191	-0.05081131	0.76660539	2.50991396
MJD	100000	4.83E-16	1.000005	-2.20110705	-0.74905506	0.154745	0.65710191	1.8487135
fiber_ID	100000	-7.69E-17	1.000005	-1.64520223	-0.83785401	-0.05986392	0.71812618	2.02089262

Cuadro II: Descripción de los datos estandarizados

mejor para cada problema, sin embargo, como primera aproximación de solución, se utilizará un modelo conocido como "$Kvecinoscercanos$". En clasificación, KNN (por sus siglas en inglés), es un algoritmo de que se basa en la proximidad de los datos de un espacio de características. Este supone que los datos que existen en una determinada aglomeración (clases en este caso) comparten ciertas características.[2]
El modelo tiene algunos hiperparámetros como la cantidad de vecinos a considerar, el tipo de distancia (Manhattan, Euclidiana) entre otros.

Comenzar con este modelo tiene varias razones, como su facilidad de entendimiento, el como maneja las aglomeraciones, su facilidad de implementación y el tipo del problema. En clasificación de objetos estelares existe el conocido diagrama de Hertzsprung-Russell, que relaciona ciertas características de las estrellas con su clase y en este se forman aglomeraciones fácilmente. [3].

C. Implementación del modelo

Una vez realizado el ETL y la selección del modelo, se puede comenzar con la implementación del modelo.

1. Separación del conjunto de datos

Para poder realizar la implementación del modelo, es necesario entrenarlo. Consecuentemente, es necesario dividir el conjunto de datos en entrenamiento y prueba. El conjunto de entrenamiento servirá para que el modelo

aprenda y reconozca patrones de los datos, ajustando los hiperparámetros de forma iterativa. Por otro lado el conjunto de entrenamiento servirá para verificar la calidad del modelo, es decir, su precisión de lo aprendido durante el entrenamiento. Para dividir los datos, se utilizará un tamaño de prueba de 20 %.

Simultáneamente, se debe de separar una parte del conjunto de entrenamiento para realizar una validación de los hiperparámetros con registros desconocidos. Este subconjunto de validación, (así como el de prueba) se toma de forma aleatoria de los registros a través de una determinada cantidad entrenamientos y se promedia el rendimiento del modelo de forma global para caracterizar la calidad del entrenamiento. A este proceso se le conoce como validación cruzada. El tamaño de este subconjunto es también de 20 % pero del conjunto de entrenamiento (16 % del total).

2. *Entrenamiento del modelo*

Para los primeros entrenamientos del modelo, solo se configuraron dos hiperparámetros, que es el tipo de distancia (p) y la cantidad de vecinos (k). Se realizo una validación cruzada de cinco entrenamientos con su respectiva verificación y se utilizó como métrica el sesgo y la varianza, calculados de acuerdo con la ecuación 1 y 2.

$$\text{Sesgo} = \bar{X}_{\text{Predicciones correctas}} - \bar{X}_{\text{Valores reales}} \quad (1)$$

$$\text{Varianza} = \sigma^2_{\text{Predicciones correctas}} \quad (2)$$

Para poder tomar estas métricas, se convirtieron a números las categorías y como resultado las predicciones correctas, son booleanos. La idea principal es medir el sobreajuste, en el cual el modelo tiene un muy buen entrenamiento dando resultados excelentes (conjunto de entrenamiento), pero es muy malo para predecir a partir de registros nuevos (conjunto de validación) o el subajuste, en el que el modelo es muy malo tanto en entrenamiento como con registros nuevos. Se dice que existe un sobreajuste para valores altos de varianza y un subajuste para valores altos de sesgo.[4] Es decir, la varianza y el sesgo son valores que se comportan de forma inversa dependiendo de la calidad del entrenamiento. En los dos primeros entrenamientos se probaron dos conjuntos de hiperparámetros uno que se consideraría simple ($p = 1, k = 1$) y otro un poco complejo ($p = 2, k = 21$). Los resultados del promedio de sesgo y varianza se muestran en la tabla III.

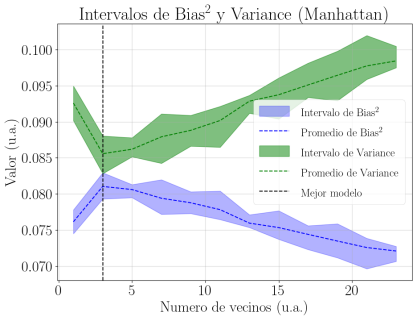
Los resultados hablan bastante por si solos. Se debe de considerar que estas métricas no están calificando realmente el desempeño del modelo, si no de su entrenamiento. En el modelo simple, el sesgo y la varianza

Hiperparámetros		**Metricas**	
p	**k**	**Sesgo**	**Varianza**
1	1	0.076336	0.072748
2	21	0.096826	0.111167

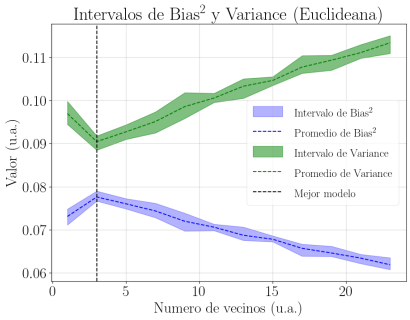
Cuadro III: Resultados de primeros dos entrenamientos.

tienen una diferencia bastante pequeña, lo que implica que ambos probablemente se encuentran alrededor de su valores mínimos por consiguiente tenemos un entrenamiento bastante bueno a pesar de su simplicidad. Por otro lado, en el modelo complejo, se puede observar que existe un sobreajuste, ya que la diferencia entre ambas métricas es bastante alta y el valor de la varianza es el mayor.

Como primeros entrenamientos, hubo resultados medianamente buenos. Sería conveniente entrenar el modelo con combinaciones de hiperparámetros en el rango de los mostrados para analizar la calidad de su entrenamiento y cuál sería el mejor modelo de acuerdo a su entrenamiento. Los resultados de dicho entrenamiento se muestran en las gráficas 1a y 1b.



(a) Sesgo y varianza para diferentes k y distancia de Manhattan ($p = 1$)



(b) Sesgo y varianza para diferentes k y distancia Euclidiana ($p = 2$)

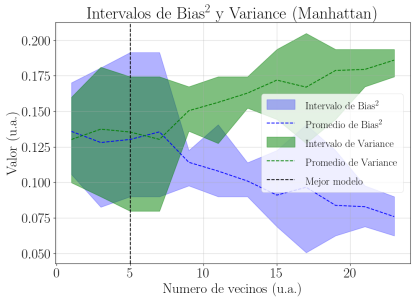
Figura 1: Sesgo y varianza para diferentes k y p

Se muestran los intervalos de sesgo y varianza que se obtuvieron a a lo largo de las validaciones y el promedio de estos. Se observa que el mejor modelo para ambos

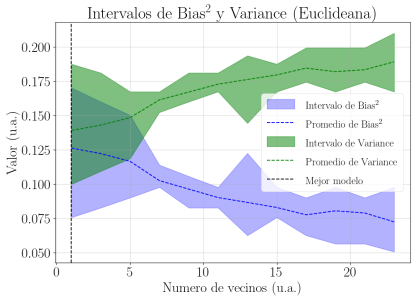
tipos de distancias, se encuentra para un $k = 3$ y que a partir de esta valor, tanto para mayores valores como menores, comienza a haber sobreajuste, ya que la varianza comienza a subir y el sesgo a bajar por lo que la distancia entre ambas métricas crece.

Si bien es cierto que para ambos valores de p tenemos comportamientos de varianza y sesgo similares, se puede destacar que para la distancia Euclidiana, los intervalos son más cortos, lo cual es mejor ya que implica que a través de las validaciones las métricas son más consistentes. Por lo tanto, el mejor modelo es $k = 3$ y $p = 2$, considerando solo lo hiperparámetros modificados.

Una cuestión destacable es el por que no se presenta subajuste en el entrenamiento. Esta cuestión se puede responder considerando algunos factores, pero el más importante es: el numero de registros. La cantidad de registros es lo suficientemente grande como para hacer que durante el entrenamiento, el modelo aprenda a reconocer los patrones lo suficientemente bien. Para probarlo, se podría reducir el tamaño de los registros de forma drástica para no dar suficiente información al modelo como para aprender y reconocer patrones. Reduciendo el tamaño de registros a quinientos (0,5 % del total) se obtienen los resultados mostrados en las gráficas 2a y 2b.



(a) Sesgo y varianza para diferentes k y distancia de Manhattan ($p = 1$)



(b) Sesgo y varianza para diferentes k y distancia Euclidiana ($p = 2$)

Figura 2: Sesgo y varianza para diferentes k y p y solo 500 registros.

Es posible observar que para la distancia Manhattan y $k = 7$, el promedio del sesgo es más grande que la varianza, por lo que se podría considerar que aunque sea mínimo, existe subajuste.

3. Prueba del modelo

Ahora que se tienen una combinación de hiperparámetros ideal, podemos probar el modelo con el conjunto de prueba. En este caso, se cambian las métricas a exactitud y la puntuación F1. En un modelo de clasificación, la exactitud, mide la proporción de clasificaciones realizadas correctamente. Por otro lado, puntuación F1 calcula el promedio ponderado entre la proporción de verdaderos positivos sobre el total de predicciones positivas y la proporción de verdaderos positivos sobre el total de valores reales positivos.[5] Las ecuaciones 3 y 4 describen estas métricas.

$$\text{Exactitud} = \frac{\text{VP} + \text{VN}}{\text{VP} + \text{VN} + \text{FP} + \text{FN}} \tag{3}$$

$$\text{Puntuación F1} = 2 \times \frac{\text{VP}}{2 \times \text{VP} + \text{FP} + \text{FN}} \tag{4}$$

Se enfatiza en el hecho de que la función para calcular la puntuación F1 tiene un hiperparámetro para el promedio ya que es un problema multiclase, la cual se establecio como *weighted* ya que las clases no están balanceadas. De igual forma, se agrega la métrica Matriz de confusión, la cual proporciona una visión de las predicciones realizadas por el modelo y las reales.

Los resultados del modelo en el conjunto de prueba se muestran en la figura 3 y la tabla IV.

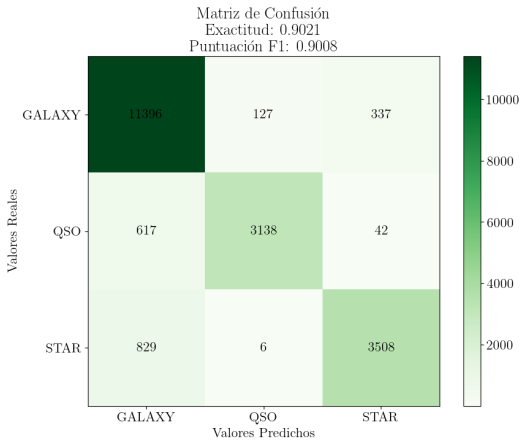


Figura 3: Matriz de confusión.

Clase	Puntuación F1	Registros
GALAXY	0.92	11860
QSO	0.89	3797
STAR	0.85	4343
Exactitud		
0.9		

Cuadro IV: Reporte de clasificación

Si bien no se trata de un modelo perfecto, se tiene un buen desempeño. La exactitud es buena y la puntuación F1 global y por clase se puede considerar alta. Debido a la alta cantidad de registros de *Galaxy* el modelo se vuelve bueno reconociendo esta clase, sin embargo, comienza a confundir a las otras clases con esta, ya que si observamos la matriz de confusión, notamos que existen bastantes *Galaxy* falsos.

D. Refinamiento del modelo

Como parte final del desarrollo alrededor del modelo, viene el refinamiento del modelo. En esta sección se hará algo parecido a lo que se hizo en la búsqueda de la mejor combinación de hiperparámetros durante la implementación, solo que en este caso, se ampliará la cantidad de hiperparámetros a refinar y el rango será más grande en algunos.

Para ello, se hará uso de una función que permite hacer la búsqueda de los mejores hiperparámetros al probar de forma aleatoria determinadas combinaciones de estos. Esta función tiene como agregado extra, el hecho de que realiza validación cruzada. A este proceso se le conoce como búsqueda aleatoria. Existe una variante, la cual busca en todas las combinaciones posibles, sin embargo, por definición tiende a ser mucho más lenta en comparación, lo cual no es conveniente a menos que se requiera con extrema necesidad.

Los hiperparámetros a refinar, así como sus respectivos valores por defecto y el rango en el que se buscarán se muestran en la tabla V.

Parametro	Defecto	Rango
n_neighbors	5	(1,23)
weights	uniform	[uniform, distance]
p	2	(1,3)

Cuadro V: Hiperparámetros del algoritmo KNN por defecto y rango de búsqueda.

La mejor combinación de hiperparámetros encontrada se muestra en la tabla VI.

n_neighbors	weights	p
6	distance	1

Cuadro VI: Mejor combinación de hiperparámetros para KNN.

Resulta interesante analizar los mejores hiperparámetros encontrados. La cantidad de vecinos se mantiene pequeña pero encontramos que el tipo de distancia resulta ser la de Manhattan. Esto último probablemente se puede explicar con el tercer hiperparámetro, el cual, con esa configuración, da mayor prioridad a los vecinos con una distancia más cercana.

Los resultados de las métricas al implementar los hiperparámetros encontrados en el modelo se muestran en la gráfica 4 y la tabla VII.Se agrego el tiempo de computación debido a que es un factor a considerar durante la implementación de un modelo si no se cuenta con mucho tiempo o recursos computacionales.

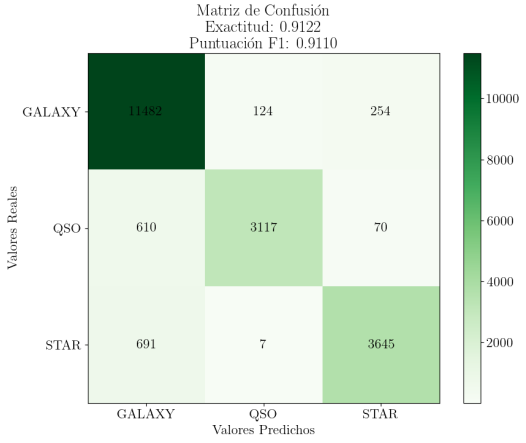


Figura 4: Matriz de confusión para KNN.

Clase	Puntuación F1	Registros
GALAXY	0.93	11860
QSO	0.88	3797
STAR	0.88	4343
Exactitud		
0.91		
Tiempo de computación		
13.66 s		

Cuadro VII: Reporte de clasificación para KNN.

Si bien no tenemos un cambio bastante grande en las métricas, existe un aumento lo que indica un refinamiento exitoso. Por otro lado, sin embargo, continuamos con los mismos inconvenientes anteriores, el modelo es muy bueno reconociendo la clase *Galaxy* pero tiende a confundir las otras clases.

E. Implementación de otros modelos

Como un agregado al desarrollo, se implementaron otros modelos de clasificación para probar diferentes opciones. A cada uno de estos modelos se les aplico un una búsqueda aleatoria para poder refinar los hiperparámetros desde el principio y utilizar la validación cruzada.

Los modelos implementados pueden ser clasificados en dos partes. La primera parte corresponde a modelos simples o fundamentales, bastante conocidos y que son la base para algoritmos o métodos más complejos. Para los primeros se utilizó Arboles de Decisión, Regresión Logística y Maquina de Soporte Vectorial. La segunda corresponde a modelos conocidos como ensambles.

Los modelos de ensamble, como su nombre indica, ensamblan modelos más simples para lograr un mejor desempeño. En especifico, se utilizaron tres de diferentes tipos. El primero, un Bosque Aleatorio, el cual ensambla Arboles de Decisión simples a través de todos los registros de forma aleatoria. A este método se le conoce como Bagging (Agregación por Bootstrap). El segundo, una Potenciación de Gradiente, el cual también ensambla Arboles de Decisión simples pero de forma iterativa con el fin de corregir los errores de los anteriores. A este se le conoce como Potenciación. El tercero y ultimo, un Clasificador por Votación. En especifico este ultimo resulta bastante interesante ya que es capaz de ensamblar diferentes modelos para utilizarlos como votantes y finalmente crear una decisión por promedio o ponderación. En este caso se utilizaron los modelos de Arboles de Decisión, Regresión Logística y Maquina de Soporte Vectorial con los hiperparámetros encontrados por búsqueda aleatoria.

Los resultados del desempeño de cada uno de estos modelos, puede ser consultado en la sección de Anexos.

III. RESULTADOS Y ANÁLISIS

Los resultados generales del rendimiento de todos los modelos aplicados se muestra en la tabla VIII.

Tipo	Modelo	Exactitud	F1	Tiempo
Simple	K vecinos	0.9122	0.911	13.66
	Regresión Logistica	0.9605	0.9601	11.21
	Arboles de Decisión	0.975	0.9748	24.16
	Maquina de Soporte Vectorial	0.9674	0.9471	116.46
Ensamble	Bosque Aleatorio	0.977	0.9768	29.3
	Potenciación de Gradiente	0.9782	0.9781	577.27
	Clasificador por Votación	0.9711	0.9709	648.51

Cuadro VIII: Desempeño final de cada uno de los modelos.

Naturalmente, observamos que los modelos de ensamble tienen un mejor desempeño, pero, también implican más tiempo de computación. Considerando tiempo y rendimiento, los modelos de Árboles de Decisión y Bosque Aleatorio son los mejores clasificando en este problema. Se observa que persiste la excelente clasificación de la clase *Galaxy* a través de los diferentes modelos a KNN pero ahora se aumenta considerablemente la clasificación de la clase *Star* siendo aún mejor. Entonces, podemos definir que el problema solo existe en la clase *Qso* ya que no se logra distinguir correctamente.

IV. CONCLUSIÓN

A lo largo de este trabajo, se logro comprender la importancia de cada uno de los pasos en la implementación de modelos de aprendizaje maquina. A primera instancia se define el porque es necesario dividir el conjunto de datos en entrenamiento, validación y prueba. Posteriormente, centrandosé en la implementación, se entiende la necesidad de medir el sesgo y la varianza durante el entrenamiento y la validación, para comenzar con ello, una búsqueda correcta de hiperparámetros. Si bien durante el refinamiento se presenta el método de búsqueda aleatoria, definir para que parámetros existe sub y sobre ajuste, permite reducir la cantidad de combinación a probar y por lo tanto, el tiempo de computación.

Finalmente, se demuestra la necesidad de probar otros modelos, tanto complejos como simples para encontrar un mejor desempeño tanto de forma global, como individual en cada clase.

[1] KDnuggets, Data transformation: Standardization vs. normalization (2020), data Science and Digital Engineering.

[2] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, 2nd ed. (Springer, New York, NY, 2009) chapter 13: Nearest Neighbors.

[3] B. W. Carroll and D. A. Ostlie, *An Introduction to Modern Astrophysics*, 2nd ed. (Pearson, 2007) chapter 19: Stellar Evolution and the Hertzsprung-Russell Diagram.

[4] A. Salazar, Sobreajuste y subajuste (overfitting and underfitting) (2019), [Azure Machine Learning].

[5] A. C. Müller and S. Guido, *Introduction to Machine Lear-*

ning with Python (O'Reilly Media, 2016).

ANEXOS

A. Repositorio

El link del repositorio donde se encuentran todos los códigos y la base datos utilizada es:

B. Desempeño de modelos

1. *Regresión Logística*

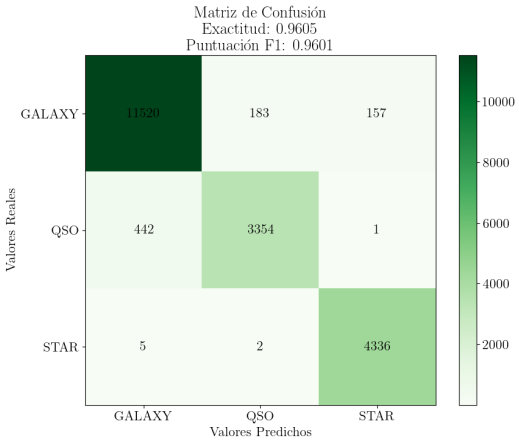


Figura 5: Matriz de confusión para Regresión Logística.

Clase	Puntuación F1	Registros
GALAXY	0.97	11860
QSO	0.91	3797
STAR	0.98	4343
Exactitud		
0.96		
Tiempo de computación		
11.21 s		

Cuadro IX: Reporte de clasificación para Regresión Logística.

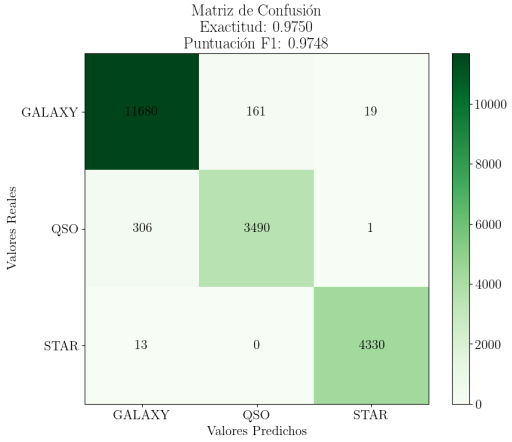


Figura 6: Matriz de confusión para Arboles de Decisión.

2. *Arboles de Decisión*

Clase	Puntuación F1	Registros
GALAXY	0.98	11860
QSO	0.94	3797
STAR	1.00	4343
Exactitud		
0.97		
Tiempo de computación		
24.16 s		

Cuadro X: Reporte de clasificación para Arboles de Decisión.

3. *Máquina de Soporte Vectorial*

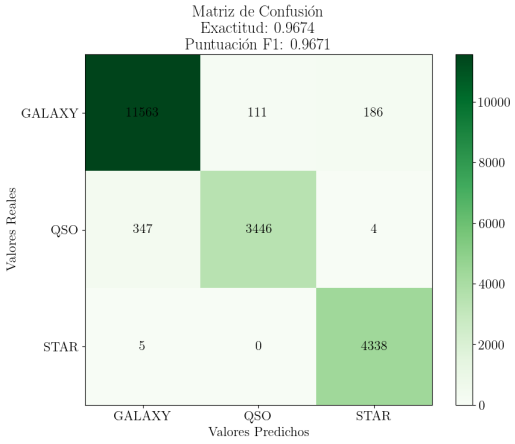


Figura 7: Matriz de confusión para Máquina de Soporte Vectorial.

Clase	Puntuación F1	Registros
GALAXY	0.97	11860
QSO	0.97	3797
STAR	0.96	4343
Exactitud		
0.97		
Tiempo de computación		
116.46 s		

Cuadro XI: Reporte de clasificación para Máquina de Soporte Vectorial.

4. *Bosque Aleatorio*

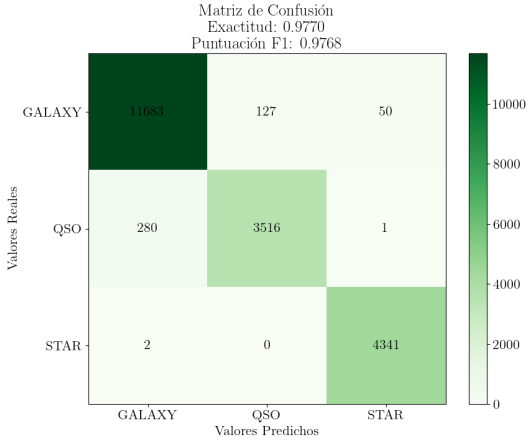


Figura 8: Matriz de confusión para Bosque Aleatorio.

Clase	Puntuación F1	Registros
GALAXY	0.98	11860
QSO	0.97	3797
STAR	0.99	4343
Exactitud		
0.98		
Tiempo de computación		
29.30 s		

Cuadro XII: Reporte de clasificación para Bosque Aleatorio.

5. *Potenciación de Gradiente*

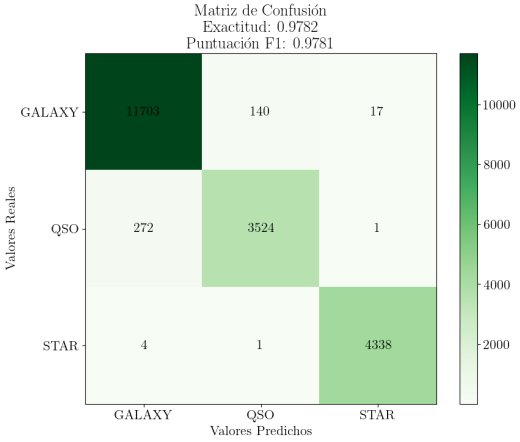


Figura 9: Matriz de confusión para Potenciación de Gradiente.

Exactitud
0.98
Tiempo de computación
577.27 s

Cuadro XIII: Reporte de clasificación para Potenciación de Gradiente.

6. *Clasificador por Votación*

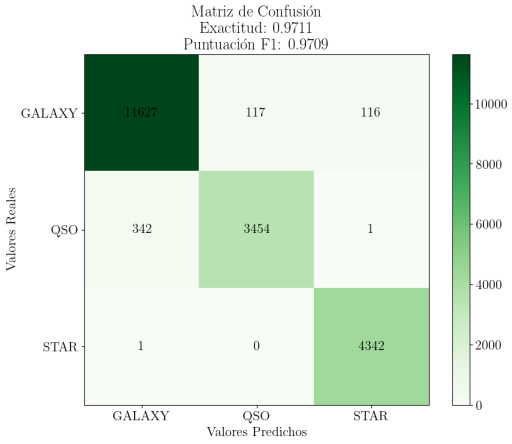


Figura 10: Matriz de confusión para Clasificador por Votación.

Clase	Puntuación F1	Registros
GALAXY	0.98	11860
QSO	0.94	3797
STAR	0.99	4343
Exactitud		
0.97		
Tiempo de computación		
648.51 s		

Cuadro XIV: Reporte de clasificación para Clasificador por Votación.