Instituto Tecnológico y de Estudios Superiores de Monterrey

Campus Estado de México

Escuela de Ingeniería y Ciencias

TC3006C.101

Inteligencia artificial avanzada para la ciencia de datos I

Momento de Retroalimentación 3: Módulo 2

Alumno:

Gustavo Alejandro Gutiérrez Valdes - A01747869

Profesor:

Jorge Adolfo Ramírez Uresti

Fecha de entrega:

Miércoles 11 de septiembre de 2024

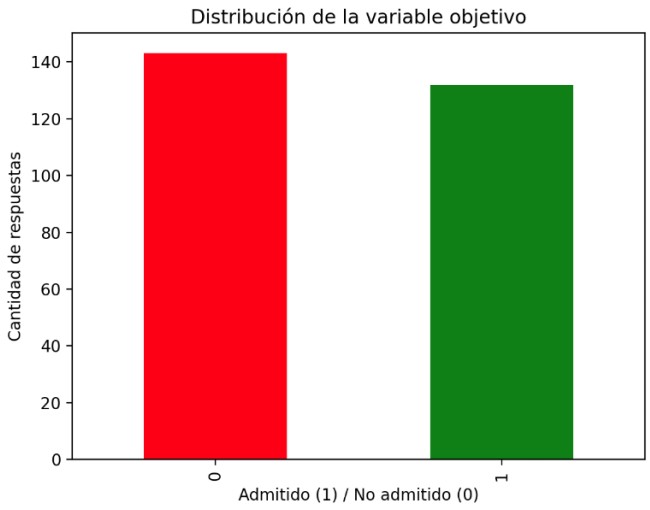
Justificación del dataset

La elección del dataset se hizo con base en la situación que se quiso replicar, la cual tiene como objetivo el predecir si un alumno será admitido en una universidad dependiendo del puntaje obtenido en el examen de admisión y su promedio acumulado en el grado educativo anterior.

Este dataset es el indicado para un modelo como RandomForestClassifier ya que cuenta con variables continúas numéricas, con las que se pueden captar relaciones no lineales entre estas y el objetivo. De igual forma, la variable objetivo es una variable binaria, por lo que el modelo de clasificación es ideal para obtener predicciones prudentes. Adicionalmente, el modelo RandomForest es adecuado porque puede modelar relaciones complejas y detectar patrones ocultos entre los datos.

Entre los datos existen algunos registros duplicados, lo que podría impactar de manera negativa y caer en un tema de "overffiting", por lo que la elección de RandomForest resulta beneficiosa para mitigar esta situación gracias a la construcción de múltiples arboles con diferentes subconjuntos de datos. El dataset cuenta con una cantidad equilibrada de registros con los dos valores de admisión, lo que ayuda de buena manera en el desempeño del modelo antes mencionado.

Para conocer el balance de clases dentro del dataset, se graficó la cantidad de valores para la variable objetivo "Admisión".



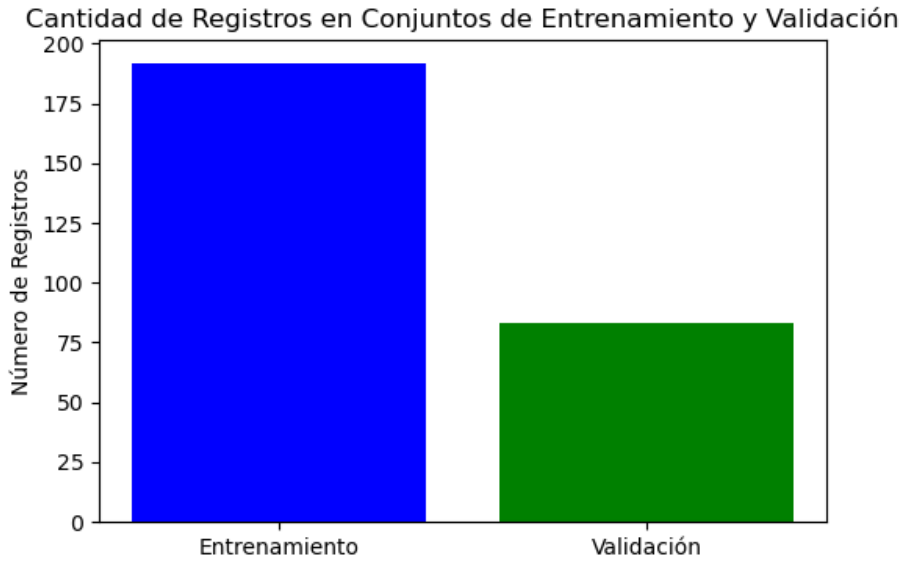
Como se puede notar, las clases se encuentra balanceadas dentro del dataset (la diferencia es mínima y no representa un impacto negativo), por lo que este conjunto de datos está conformado correctamente para poder entrenar un modelo sin tener consecuencias en las predicciones hechas y tener un resultado óptimo en la generalización con nuevos datos. Igualmente hay una gran diversidad de ejemplos por combinaciones (Puntaje del examen y Promedio Educativo) que fortalecen la capacidad de generalización del modelo.

Separación y evaluación del modelo con un conjunto de prueba y un conjunto de validación (Train/Test/Validation).

Para la separación del dataset para las etapas de entrenamiento y validación, se utilizó una función propia del framework con la que se determinó que, del total de registros disponibles, se destinarían el 70% para el entrenamiento y el restante para la etapa de validación.

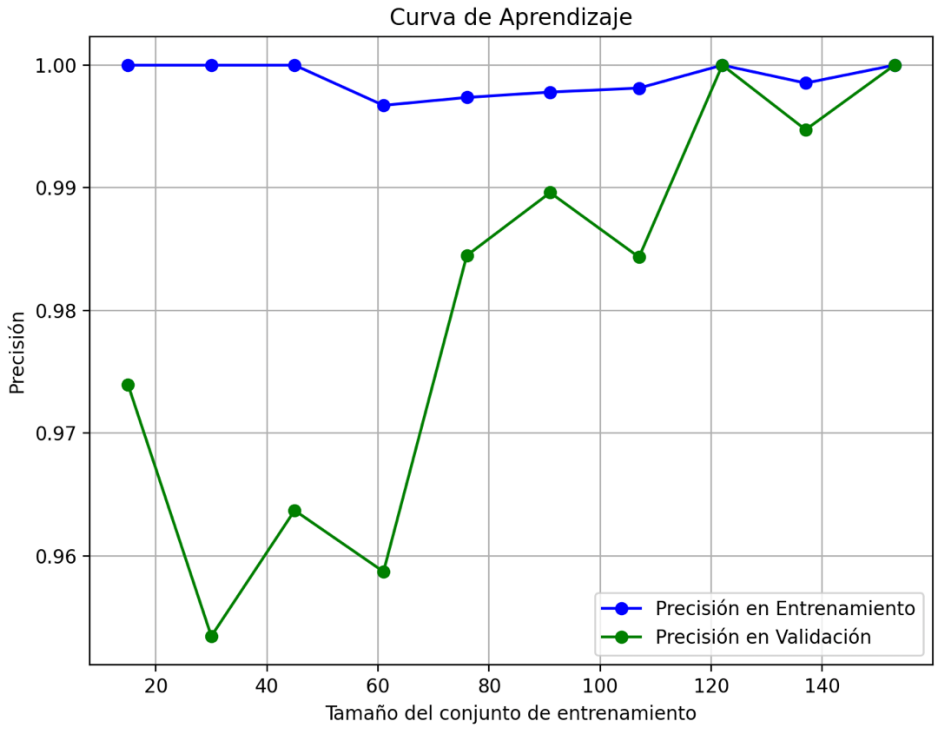
```
# Dividir en entrenamiento y validación (70% train, 30% validation)
X_train, X_val, y_train, y_val = train_test_split(features, objective, test_size=0.3, random_state=35)
```

El total de registros dentro del dataset es de 275, los cuales quedaron distribuidos de la siguiente manera:



En cuanto al dataset para las predicciones (test), se generó otro archivo *csv* con 25 registros de los cuales no se conoce el valor real, y se probará el modelo para conocer como sería su desempeño en un ejemplo más real (datos desconocidos y de los que no se tiene la respuesta).

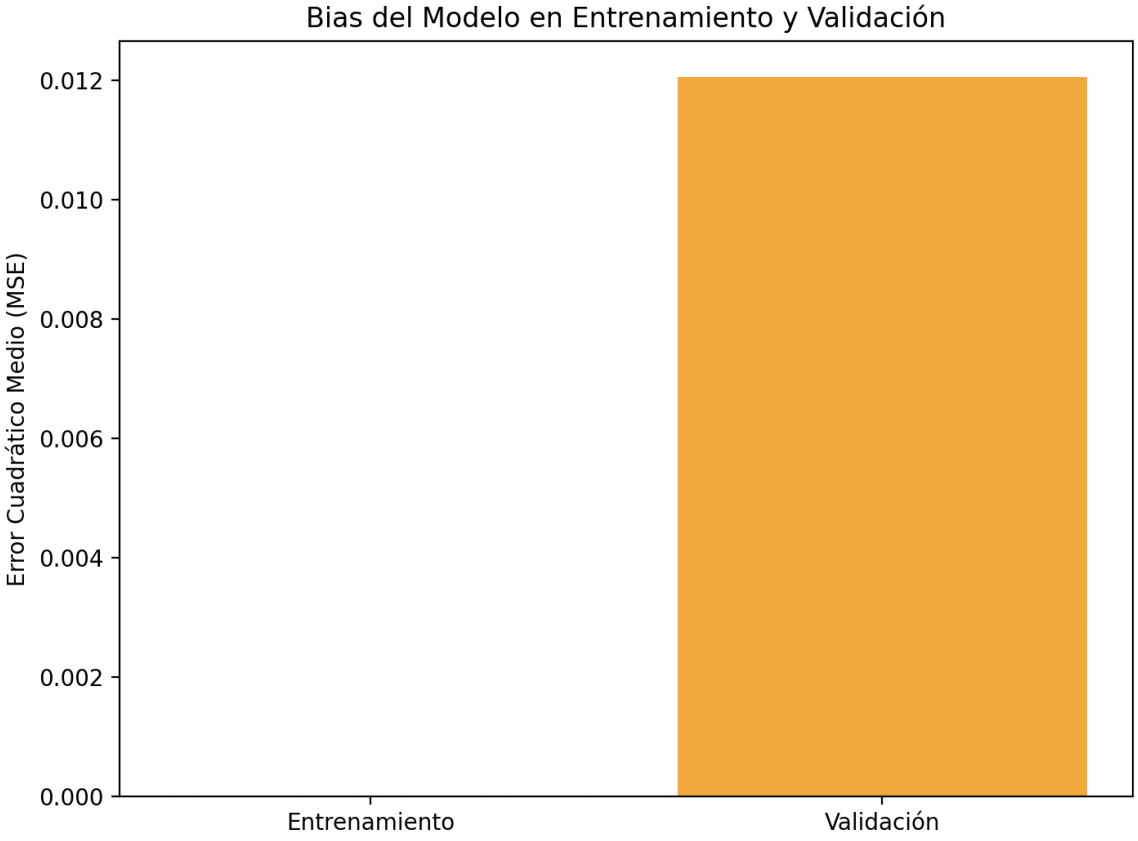
Diagnóstico y explicación el nivel de ajuste del modelo: *underfitt, fitt u overfitt*



Utilizando esta gráfica, podemos identificar el estado del modelo en cuanto a si esta *overfitt, fitt o underfitt*. Para interpretar de manera correcta esta imagen, debemos saber el significado de la precisión conforme aumenta el conjunto de datos de entrenamiento. Si la precisión de "train" se mantiene alta pero la correspondiente de la etapa de validación es baja, significa que el modelo está en estado de *overfitt*. Por otra parte, si las dos precisiones se mantienen bajas, quiere decir que el modelo esta se encuentra en

underfitt. Finalmente, como podemos notar, la precisión en entrenamiento es bastante alta, y la misma de la etapa de validación se mantiene en valores altos (difiriendo en el peor punto por 0.05), lo que significa que el modelo está en estado "*fitt*".

Diagnóstico y explicación el grado de bias o sesgo: bajo medio alto

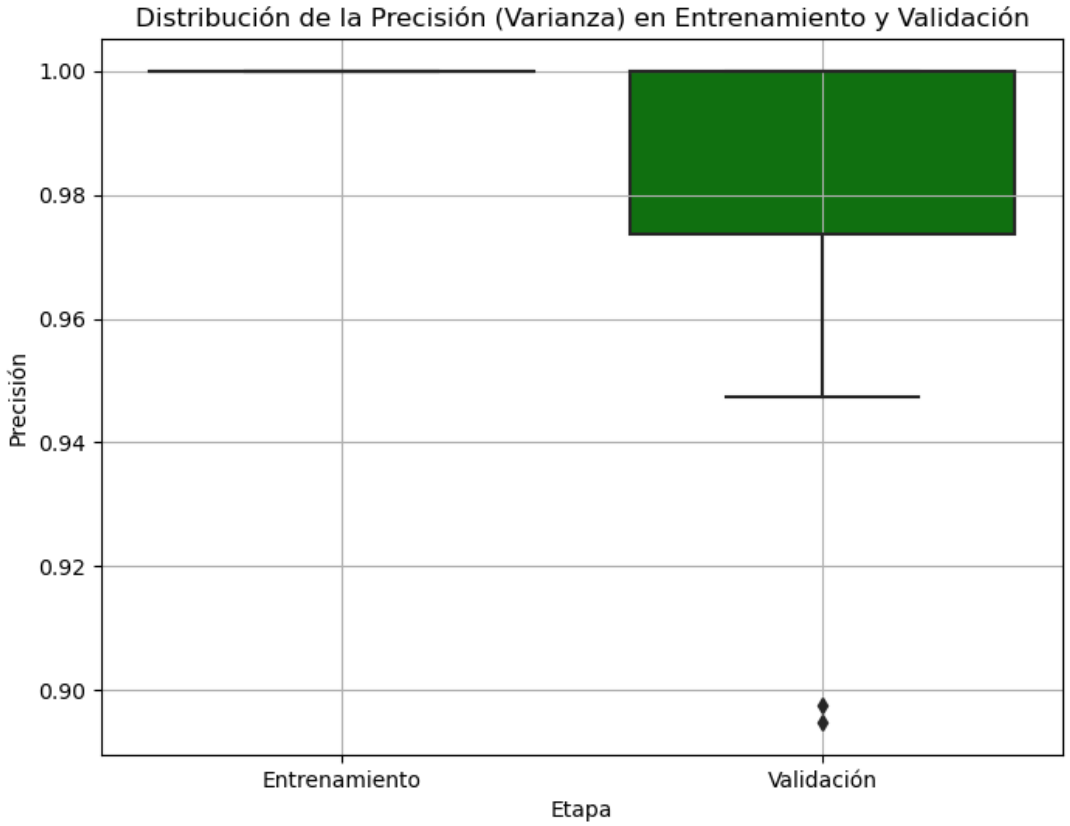


Esta gráfica compara el error cuadrático medio (*MSE*) de los conjuntos de entrenamiento y validación, lo cual es una pista esencial para calcular el *bias* o sesgo del modelo. Esta métrica nos indica que tan lejos están las predicciones del modelo con respecto a los valores reales. En caso de que el *MSE* fuera muy alto en el conjunto de entrenamiento, esto indicaría un alto *bias* que sería el reflejo de que el modelo no esta capturando bien las relaciones entre datos, que podría caer en un estado de *underfitting*. En caso contrario,

si el MSE fuera alto en el conjunto de validación, significaría que el modelo se encuentra en estado de *overfitting* ya que está fallando al generalizar con nuevos datos.

Como se puede observar, el *MSE* en la etapa de entrenamiento es 0, mientras que este valor para la etapa de validación es muy pequeño ya que tiene un valor de 0.012, por lo que se puede notar que el bias o sesgo es muy reducido y el modelo ha tenido un buen desempeño al generalizar con datos nuevos.

Diagnóstico y explicación el grado de varianza: bajo medio alto



Para el cálculo de la varianza, se hará uso de un diagrama de caja que nos ayuda a comparar las distribuciones de las métricas (en este caso la precisión) para las etapas de entrenamiento y validación. Nos permite visualizar la mediana, los cuartiles y los valores atípicos de las métricas analizadas.

Como se puede observar en la gráfica, la dispersión de la precisión en la etapa de entrenamiento es nula, mientras que en la de validación es bastante pequeña tomando en cuenta la escala de la gráfica, teniendo su valor mínimo aproximadamente en 0.95 y teniendo los valores atípicos aproximadamente en 0.88. Por lo tanto, se puede concluir que la varianza es bastante baja en el modelo y, por ende, el desempeño del modelo es más estable y predecible.

Basándote en lo encontrado en tu análisis utiliza técnicas de regularización o ajuste de parámetros para mejorar el desempeño de tu modelo y documenta en tu reporte cómo mejoró este

Para el manejo de hiperparámetros, se utilizó la función de *grid search* con el objetivo de obtener la mejor combinación de estos posibles para posteriormente dársela al modelo y que se entrene con ellos. Por lo tanto, es muy complicado encontrar mejoras substanciales en el desempeño del modelo, ya que la precisión se encuentra en un 99%, lo que nos habla de un rendimiento excelente por parte del modelo.

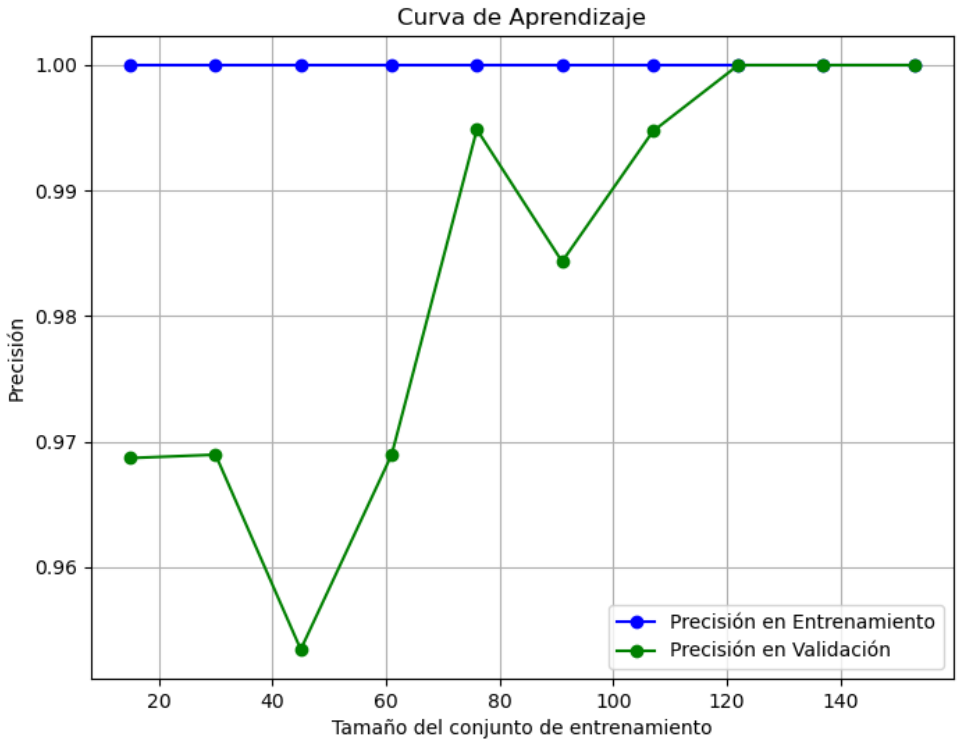
```
# Se definen los diferentes conjuntos de hiperparámetros que se probarán entre si para encontrar la mejor combinación en el
# siguiente paso. Se eligieron la cantidad de arboles a generar, la profundiad máxima de cada uno y la cantidad  de muestras
# con la que los nodos se dividirán, ya que estos hiperparámetros controlan directamente la capacidad predictiva al
# reducir la varianza y hacer las predicciones mas robustas. De igual forma, se mantiene dominio acerca del subajuste o
# sobreajuste del modelo y finalmente se mitiga el impacto del ruido, dandole al modelo una mayor capacidad de generalización.
param_grid = {
    'n_estimators': [10, 50,100, 150],
    'max_depth': [None, 5, 10,20],
    'min_samples_split': [2, 5, 10]
}

# Crear un objeto GridSearchCV. Este objeto se encargará de probar todas las combinaciones posibles de hiperparámetros y elegir
# la que mejor funcione con el modelo elegido, se probará con validación cruzada de 5 folds y se utilizarán todos los núcleos
# disponibles para el trabajo
grid_search = GridSearchCV(estimator=RandomForestClassifier(),
                           param_grid=param_grid,
                           cv=5,
                           n_jobs=-1)
```

Las siguientes métricas son pertenecientes al desempeño del modelo actual, y nos servirán para poder demostrar si es que se pudo obtener una mejora significativa en estas cantidades con las modificaciones que haremos a continuación.

Reporte de clasificación:

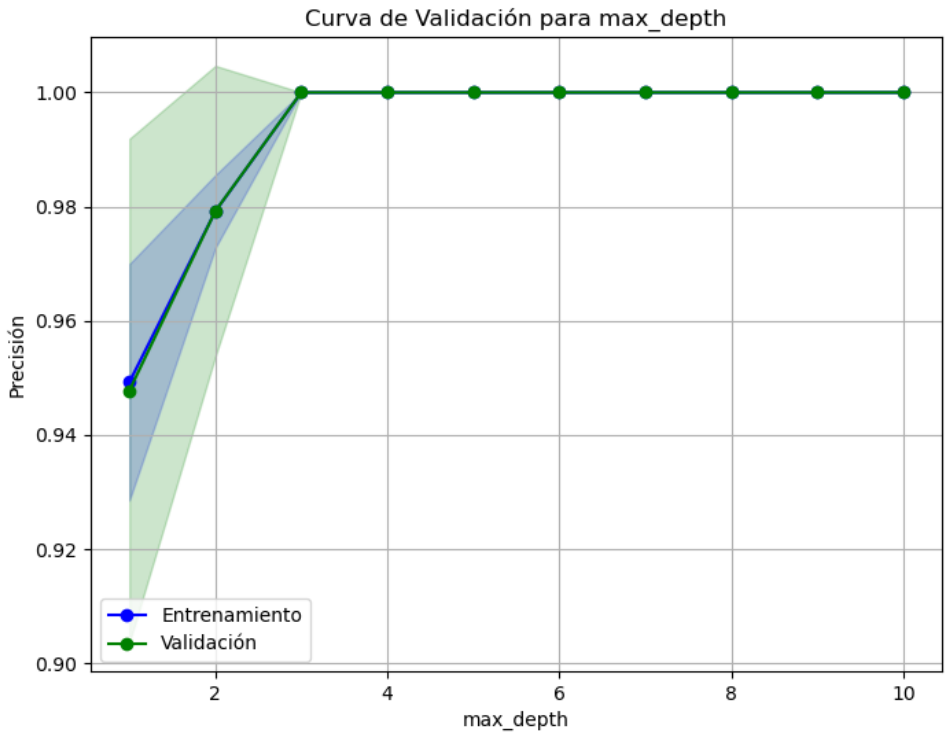
	precision	recall	f1-score	support
0	0.98	1.00	0.99	46
1	1.00	0.97	0.99	37
accuracy			0.99	83
macro avg	0.99	0.99	0.99	83
weighted avg	0.99	0.99	0.99	83

Como primera técnica de mejora, se modificará el valor del número de árboles (*n_estimators*) buscando mejorar la estabilidad del modelo. Igualmente se tiene la posibilidad de reducir la varianza de las métricas, aunque se sacrifica un tiempo más prolongado de entrenamiento.



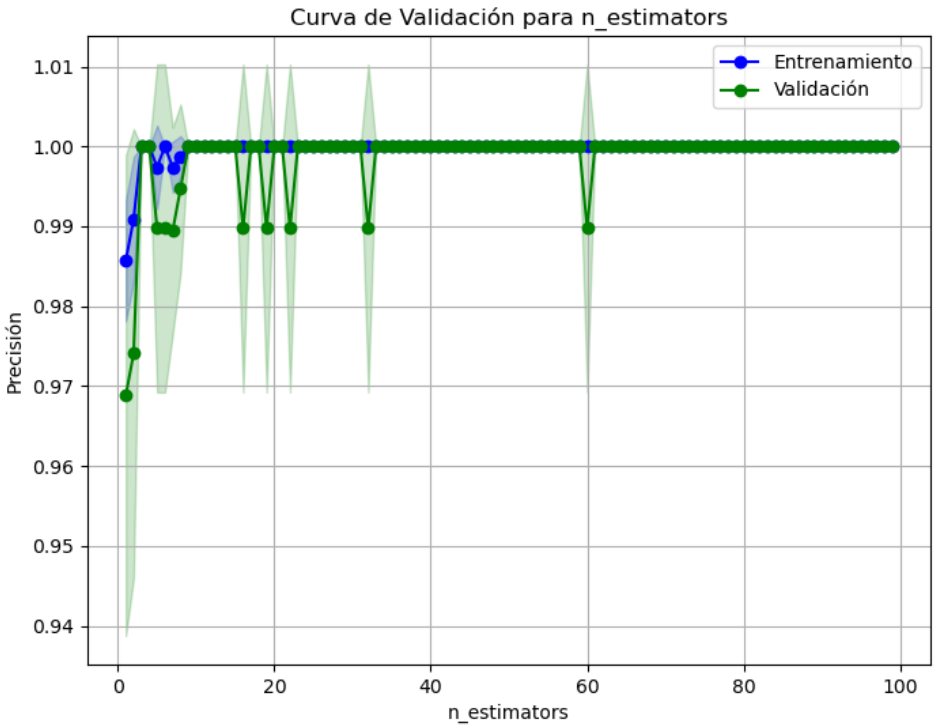
Como resultado, en cuanto a métricas se obtuvieron las mismas cantidades, pero se puede notar un comportamiento menos errático en cuanto a la precisión tanto en la etapa de entrenamiento como de validación en la curva de aprendizaje del modelo, lo que podemos considerar como un desempeño más estable por parte de este último.

Continuando con la siguiente técnica, se realizó una curva de validación de dos hiperparámetros del modelo, max_depth y n_estimators. Esto se hizo con el objetivo de medir el impacto de estos en el desempeño del modelo, y con base en la precisión de cada uno, se podrá saber si se deben modificar. La gráfica obtenida fue la siguiente:



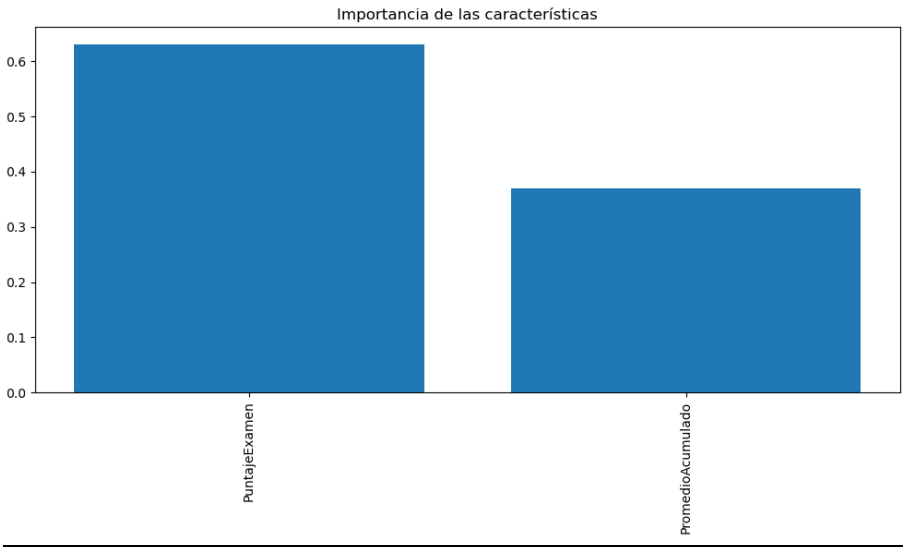
Esta primera gráfica corresponde a la *max_depth*, en donde se puede observar que la precisión en las dos etapas es muy alta, y que convergen las dos líneas graficadas por lo que se puede concluir que el modelo tiene un buen balance entre capacidad de ajuste y generalización, por lo que no se requeriría de mas ajuste de este hiperparámetro.

Siguiendo con el siguiente hiperparámetro, *n_estimators,* se obtuvo la siguiente gráfica:



Aquí se puede observar un comportamiento más errático al inicio por parte de la precisión con relación con el hiperparámetro, pero no se nota una caída considerable de esta métrica. Además, se tuvo que restringir el rango de valores hasta 100 debido a temas de procesamiento, pero finalmente se puede identificar que las dos líneas convergen en su mayoría, por lo que el desempeño del modelo con estos valores es bastante bueno.

Por último, para asegurarnos que los pesos de las características no son tan desbalanceados en el desempeño del modelo, se realizó una gráfica para conocer el peso específico de estas. El resultado obtenido fue el siguiente:



Esta gráfica se obtuvo por medio de una función de la librería de scikit-learn, *best_model.feature_importances_*, el cual devuelve un arreglo con la importancia de cada característica en el modelo. Esta se calcula con base en cuanto contribuyen las características a la predicción del modelo. En cuanto a RandomForest, esto se mide a la reducción media de impureza (*Gini*) en los nodos de todos los árboles resultantes a la división de datos. La importancia se mide en un rango de 0 a 1, por lo que se puede observar que el valor del Puntaje del examen es mayor al Promedio Acumulado, pero no es una diferencia tan exagerada, por lo nos das otra perspectiva o explicación del funcionamiento tan correcto del modelo en las etapas de evaluación.

Por lo tanto, se puede concluir que el funcionamiento del modelo de predicción es el esperado, y se tiene una confianza bastante grande en cuanto a sus resultados. Tanto la elección de los datos (entrenamiento y validación), así como el modelo con sus hiperparametros tienen un impacto positivo en los resultados arrojados por el modelo, y se tiene una precisión casi perfecta que demuestra que el proceso de desarrollo de este fue satisfactorio para la actividad.