Instituto Tecnológico de Estudios Superiores de Monterrey

Campus Estado de México

Escuela de Ingeniería y Ciencias
Ingeniería en Ciencias de Datos y Matemáticas

Concentración en Inteligencia Artificial Avanzada

Momento de Retroalimentación: Módulo 2 Análisis y Reporte sobre el desempeño del modelo

Diego Fernando Landeros Austria

A01751654

Profesor: Jorge Adolfo Ramírez Uresti

Fecha de elaboración:

11 de septiembre del 2023

Diego Fernando Landeros Austria A01751654
Momento de Retroalimentación: Módulo 2

Índice

Diego Fernando Landeros Austria A01751654
Momento de Retroalimentación: Módulo 2

I. Introducción

En este trabaja se realiza un análisis del desempeño de un modelo de clasificación usando Gradient Boosting Classifier sobre un conjunto de datos que busca predecir si un individuo sufrirá un derrame basado en variables como el género, la edad, el nivel promedio de glucosa, el índice de masa corporal, si se ha casado, si sufre hipertensión o enfermedades cardiacas, entre otros. Para realizar el análisis se realizará un conjunto de diagnósticos variados que sirven para explicar el comportamiento del modelo e identificar posibles problemas que puedan existir en el modelo. A continuación se enlistan las pruebas a las que se someterá al modelo: Diagnóstico del grado de bias, diagnóstico del grado de varianza, diagnóstico del nivel de ajuste del modelo. Adicionalmente, se trata la división del modelo en conjuntos de entrenamiento, prueba y validación y se utilizan técnicas de regularización y ajuste de parámetros para mejorar el desempeño del modelo.

II. Justificación de elección del dataset

A partir de la exploración de diferentes datasets, principalmente usando la plataforma kaggle, se decidió por un dataset llamado “Healthcare Dataset Stroke Data”, este es usado para predecir si un paciente es propenso a sufrir de un derrame cerebral basado en diferentes parámetros que caracterizan la información relevante de una persona.

Se eligió trabajar en este dataset principalmente por dos razones: La primera razón es la alta calidad de los datos, usando el filtro de ‘usability rating’ en kaggle se acotó la búsqueda a datasets de alta calidad, dado que el objetivo principal de este trabajo no es la limpieza de los datos siempre es preferible utilizar datos limpios y de alta calidad para obtener mejores resultados en el modelo. La segunda razón para elegir el dataset fue el área de los datos, escoger un dataset relacionado con el área de la salud nos permite imaginar de mejor manera la posible utilización del modelo y, en este caso, entender el significado de las variables de manera fácil, además que los estudios existentes del tema nos permiten cotejar si el ajuste del modelo es congruente con los factores de riesgo conocidos.

A continuación se realiza una exploración de las variables y su significado:

- ‘sex’ : El género del paciente. 1 corresponde a masculino y 0 a femenino
- ‘age’: La edad del paciente en años
- ‘hypertension’: Indica si el paciente alguna vez ha padecido hipertensión (1) o no (0)
- ‘heart_disease’: Indica si el paciente ha padecido enfermedades cardiacas (1) o no (0)
- ‘ever_married’: Indica si el paciente se ha casado (1) o no (0)
- ‘work_type’: Indice el tipo de trabajo del paciente, se categoriza como nunca ha trabajado (0), niños (1), empleado del gobierno (2), empleado autónomo (3) o trabajador privado (4)
- ‘Residence_type’: Indica si el paciente vive en un área urbana (1) o rural (0)
- ‘avg_glucose_level’: Indica el nivel promedio de glucosa en la sangre de cada paciente

- 'bmi': Es el índice de masa corporal de cada paciente
- 'smoking_status': Indica si el paciente fuma (1) o nunca ha fumado (0)
- 'stroke': Es nuestra variable a predecir, nos dice si el paciente es propenso a sufrir un derrame cerebral (1) o no (0)

Es importante comentar que para evitar tiempos de entrenamiento muy largos y por poder de cómputo se acorta el conjunto de datos para tener 5,000 muestras, óptimamente usamos todos los datos para un mejor aprendizaje del modelo.

III. Separación y evaluación del modelo con un conjunto de prueba y un conjunto de validación

La separación de los datos en 3 conjuntos (entrenamiento, validación y prueba) nos ayuda a evitar el overfitting y underfitting en el modelo.
Para hacer esta división de los datos usamos las siguientes líneas de código:

```python
X_train, X_, y_train, y_ = train_test_split(X, y, test_size=0.3, random_state=6)
X_test, X_val, y_test, y_val = train_test_split(X_, y_, test_size=0.5, random_state=6)
```

Figura 1. Código para la división del dataset en conjuntos de entrenamiento, validación y prueba.

A continuación se presenta la comparación del tamaño en los conjuntos de datos usando solamente conjuntos de entrenamiento y prueba a comparación de usar conjuntos de entrenamiento, prueba y validación:

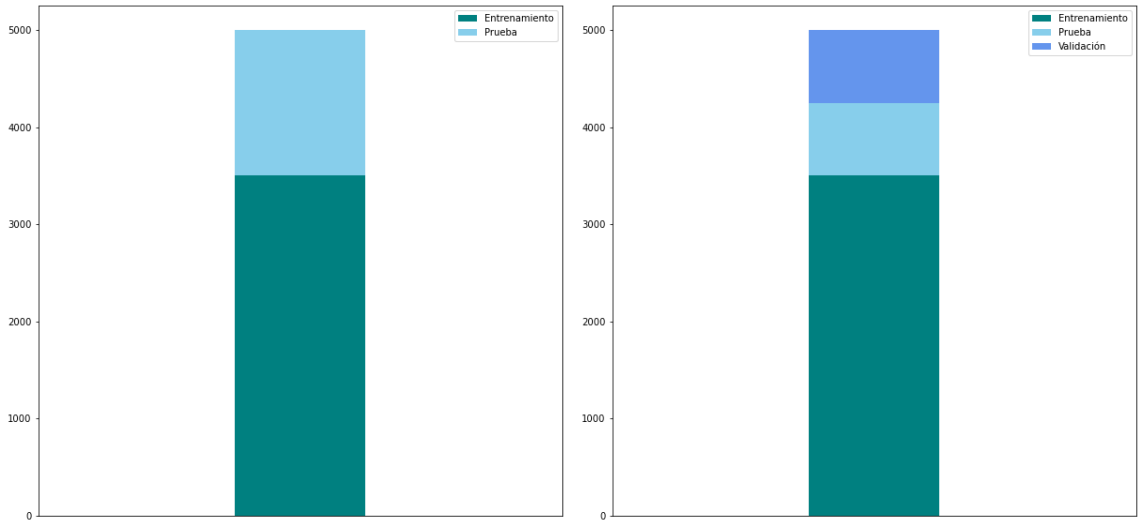


Figura 2 y 3. Proporción de los conjuntos, dividiendo en prueba y entrenamiento (2) e incluyendo la división en validación (3).

Podemos ver que la mayoría de los datos se encuentran todavía en el conjunto de entrenamiento, ya que de aquí es donde el modelo tiene que aprender las relaciones en los datos, por otro lado al introducir el conjunto de validación, estamos dividiendo el tamaño del

conjunto de prueba en 2. De esta manera podemos evaluar el modelo con el conjunto de validación, ajustarlo y utilizar el conjunto de prueba para evaluar el posible rendimiento del modelo con otros datos desconocidos.

A continuación, se selecciona una variable para visualizar cómo se hace la división de conjuntos, la división se hace de manera pseudo aleatoria (se utiliza un random state para hacer repetible el proceso), de esta manera logramos tener representatividad de todas las clases y en variedad de valores para las variables. Utilizamos la variable “avg glucose level” por su naturaleza continua y hacemos un scatter plot con los puntos coloreados de manera que correspondan con el conjunto al que pertenecen

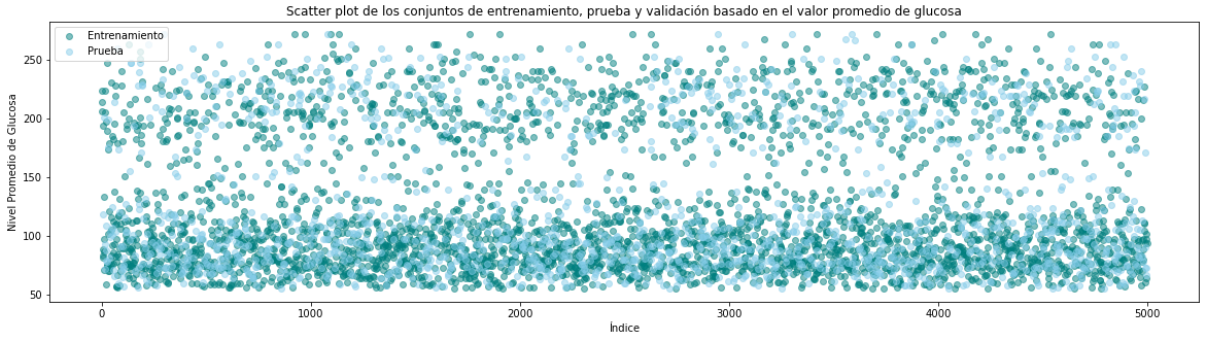


Figura 4. Distribución de los datos con una variable de ejemplo en la división en conjuntos de entrenamiento y prueba

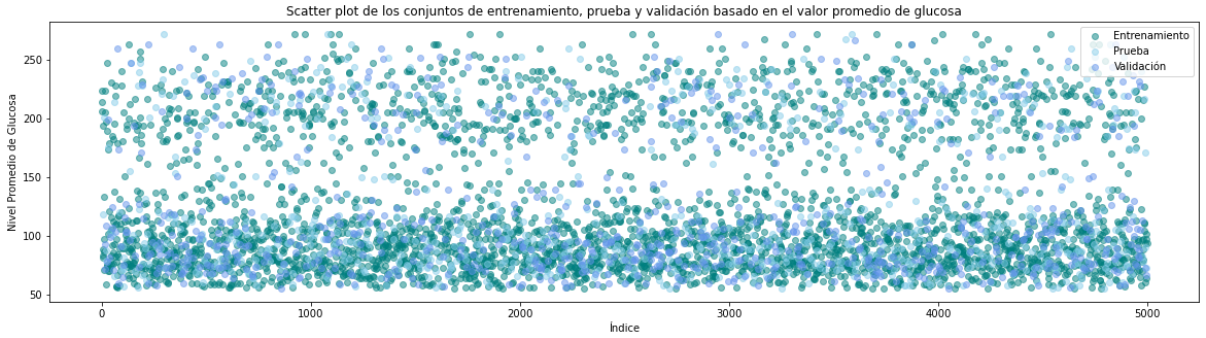


Figura 5. Distribución de los datos con una variable de ejemplo en la división incluyendo el conjunto de validación

Posteriormente, comparemos algunos valores de estadística básica en los 3 conjuntos para observar que son similares, al ser muestras del dataset pero no son idénticos al tomar diferentes partes del dataset:

	Entrenamiento	Prueba	Validación
# de elementos	3,500	750	750
Promedio	122.25	120.732	120.818
Desviación estándar	58.043	56.368	57.091
Mínimo	55.22	55.22	55.250000
Cuartiles	25%: 77.945000 50%: 98.020000	25%: 78.990000 50%: 96.970000	25%: 78.830000 50%: 97.130000

	75%: 171.230000	75%: 154.930000	75%: 161.337500
Máximo	271.74	271.74	271.74

Figura 6. Estadística descriptiva de los conjuntos de entrenamiento, validación y prueba

Finalmente, revisamos los primeros 10 índices de cada conjunto para ver que son diferentes partes del dataset:

- Conjunto de entrenamiento: [0, 1, 2, 3, 4, 6, 7, 8, 9, 11]
- Conjunto de prueba: [5, 10, 20, 21, 30, 37, 47, 57, 60, 83]
- Conjunto de validación: [14, 16, 18, 25, 29, 32, 40, 48, 50, 52]

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

Para poder calcular y definir el sesgo en el modelo que se ha entrenado, seguimos varios pasos, primeramente se usa la siguiente línea de código para descomponer el sesgo y la varianza del modelo.

```python
loss, bias, var = bias_variance_decomp(gbc, X_train_, y_train_,
X_test_, y_test_, loss='0-1_loss', num_rounds=5, random_seed=1)
```

Figura 7. Código para realizar la descomposición del sesgo y la varianza del modelo

Gracias a esta función obtenemos los siguientes resultados:

- Pérdida: 0.2203
- Sesgo: 0.2173
- Varianza: 0.0648

Además usamos la siguiente línea de código para hacer las curvas de aprendizaje del modelo, esto significa obtener el rendimiento del modelo en los conjuntos de entrenamiento y validación a medida que aumenta el tamaño del conjunto de entrenamiento. El análisis del rendimiento del modelo en los conjuntos de entrenamiento y prueba.

```python
train_sizes, train_scores, test_scores = learning_curve(gbc, X, y)
```

Figura 7. Código para realizar las curvas de aprendizaje

A partir de esto, obtenemos la siguiente gráfica:

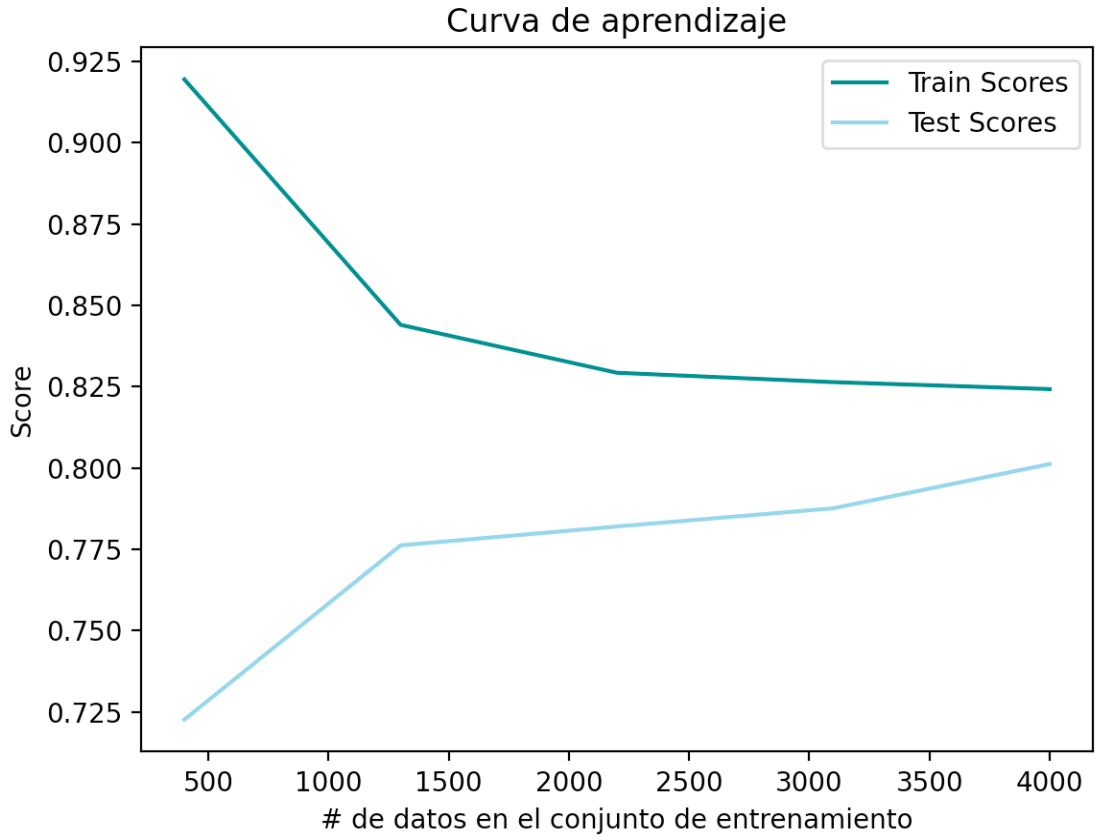


Figura 8. Curva de aprendizaje

Interpretando estos resultados, observamos que la pérdida es baja con un valor de 0.22 y el sesgo es bajo con un valor de 0.2173. El valor de sesgo que devuelve esta función nos indica si el modelo está realizando underfitting. Al ser un valor bajo de sesgo podemos decir que el modelo se está ajustando de buena manera a los datos.

Finalmente, usamos la curva ROC, para evaluar el modelo en la clasificación, esta nos aporta para explicar el bajo nivel de varianza del modelo diagnosticado observando que la curva se encuentra lejos de una diagonal, que representaría un modelo aleatorio. Además, revisando el valor de AUC que obtenemos podemos observar que ninguno es bajo, lo que indica un buen ajuste del modelo.

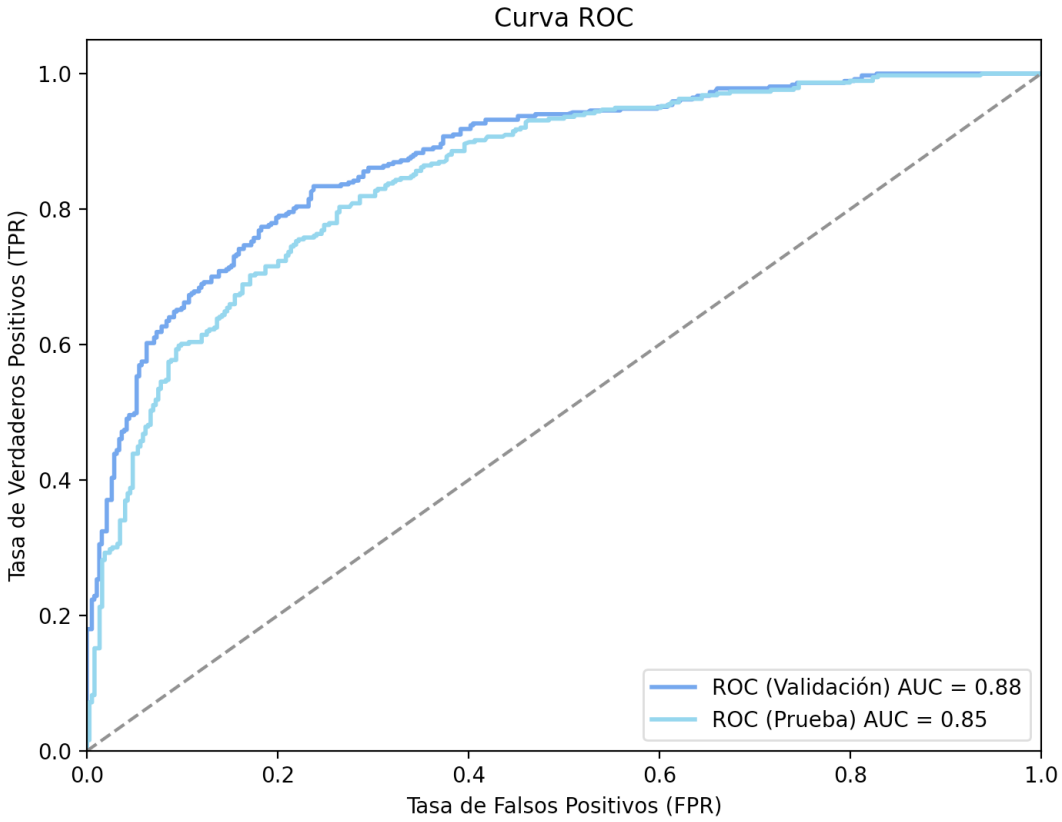


Figura 9. Curva ROC del modelo en los conjuntos de validación y prueba

V. Diagnóstico y explicación el grado de varianza: bajo medio alto

A continuación, para detectar el grado de varianza retomamos la función de la figura 7 y la gráfica en la figura 8., donde la función nos devuelve un valor de varianza de 0.0648, lo que indica una varianza baja y que el modelo no tiene alta sensibilidad a las variaciones en los datos de entrenamiento. Sin embargo, la alta variabilidad de los puntajes en la curva de aprendizaje nos indica alta variabilidad, esto es contradictorio a lo que apuntan otras métricas por lo que debemos utilizar otras pruebas para confirmar el grado de varianza del modelo.

Finalmente, retomando la curva ROC que se realizó y observando que no existen diferencia significativas en las curvas ROC de los conjuntos de validación y prueba, ni en el valor de AUC podemos descartar un alto grado de varianza en el modelo y decir que se está ajustando correctamente.

VI. Diagnóstico y explicación el nivel de ajuste del modelo: underfit, fit, overfit

Tomando en cuenta el resultado de la curva de aprendizaje, los resultados de la función usada para descomponer la varianza y sesgo, las curvas ROC y aumentando los resultados de la evaluación del modelo en los conjuntos de validación y prueba con las siguientes métricas presentadas a continuación, podemos llegar a una conclusión.

	Validación	**Prueba**
Accuracy	82.53%	78.67%
Recall	80.93%	78.46%
Precisión	82.96%	78.88%
F1 Score	81.93%	78.67%
Confusion Matrix	[322 61] [70 297]	[295 79] [81 295]

Figura 10. Métricas de evaluación del modelo en los conjuntos de validación y prueba

El modelo entrenado tiene un buen ajuste, captura bien las relaciones en los datos pero tiene un ligero sobreajuste, ya que el puntaje en el conjunto de validación es superior al de prueba, pero el sobreajuste que se presenta es menor y no presenta un problema grave en el modelo. Sin embargo, el modelo tiene áreas de oportunidad en las que puede mejorar, por ejemplo se puede mejorar el puntaje del modelo en ambos conjuntos o disminuir la diferencia en el desempeño del modelo en los datos de prueba.

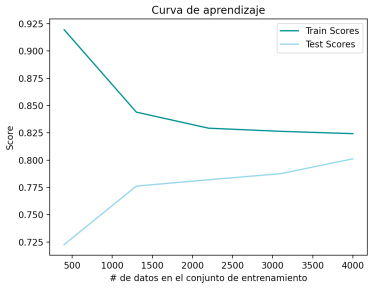
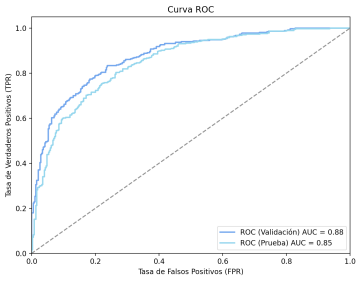
VII. Uso de técnicas de regularización o ajuste de parámetros para mejorar el desempeño

A continuación se buscará mejorar el desempeño y ajuste del modelo usando diferentes parámetros y técnicas de regularización, se modifican los parámetros del modelo de número de estimadores y learning rate y subsample. Se observará el comportamiento del modelo y finalmente, se presentará un modelo combinando estás modificaciones intentando llegar al mejor modelo.

	Validación	**Prueba**
Modelo original	Accuracy: 0.8253 Recall: 0.8093 Precision: 0.8296 F1 Score: 0.8193 [322 61] [70 297]	Accuracy: 0.7867 Recall: 0.7846 Precision: 0.7888 F1 Score: 0.7867 [295 79] [81 295]
Número de estimadores =200	Accuracy: 0.8813 Recall: 0.8856 Precision: 0.8737 F1 Score: 0.8796 [336 47] [42 325]]	Accuracy: 0.8533 Recall: 0.8590 Precision: 0.8500 F1 Score: 0.8545 [317 57] [53 323]]

Número de estimadores = 50	Accuracy: 0.7693 Recall: 0.7084 Precisionn: 0.7975 F1 Score: 0.7504 [317 66] [107 260]]	Accuracy: 0.7293 Recall: 0.6888 Precision: 0.7507 F1 Score: 0.7184 [288 86] [117 259]]
Learning Rate =0.05	Accuracy: 0.7747 Recall: 0.7084 Precision: 0.8075 F1 Score: 0.7547 [321 62] [107 260]	Accuracy: 0.7333 Recall: 0.6968 Precision: 0.7529 F1 Score: 0.7238 [288 86] [114 262]]
Learning Rate = 0.2	Accuracy: 0.8893 Recall: 0.9101 Precision: 0.8698 F1 Score: 0.8895 [333 50] [33 334]	Accuracy: 0.8613 Recall: 0.8883 Precision: 0.8434 F1 Score: 0.8653 [312 62] [42 334]
Subsample = 0.8	Accuracy: 0.8587 Recall: 0.8447 Precisionn: 0.8635 F1 Score: 0.8540 [334 49] [57 310]	Accuracy: 0.7987 Recall: 0.8005 Precision: 0.7984 F1 Score: 0.7995 [298 76] [75 301]

Figura 11. Comparación de las métricas de evaluación del modelo en los conjuntos de validación y prueba con diferentes ajustes de parámetros

	Curva de aprendizaje	Curvas ROC
Modelo original		

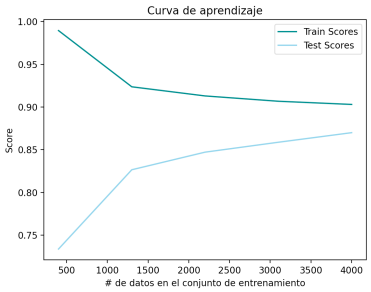
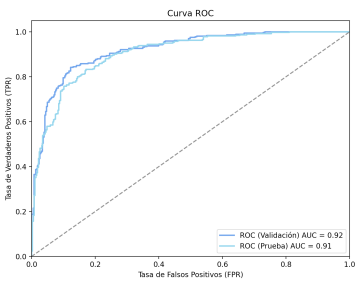
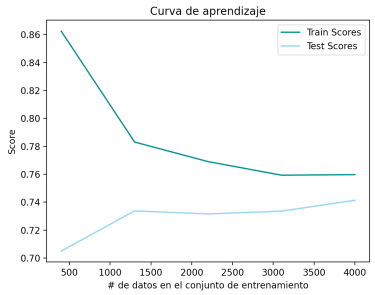
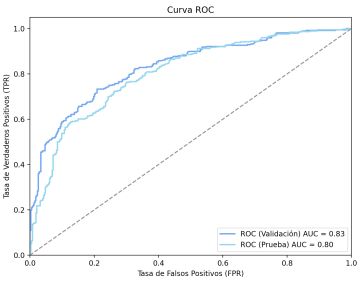
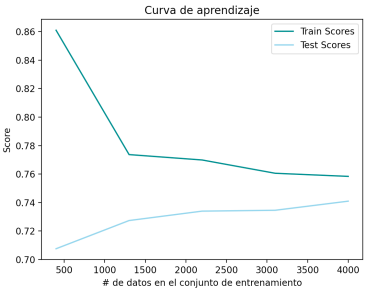
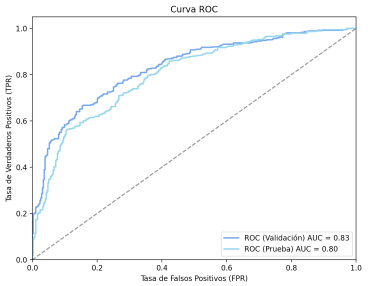
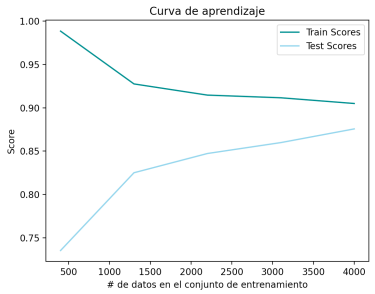
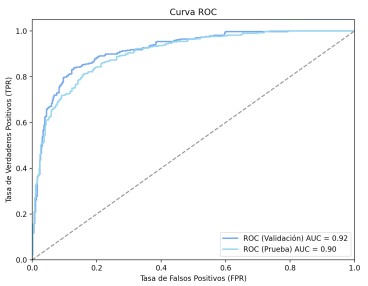
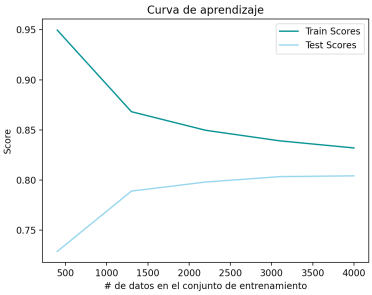
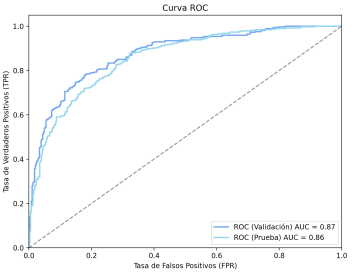
Número de estimadores =200		
Número de estimadores = 50		
Learning Rate =0.05		
Learning Rate = 0.2		
Subsample = 0.8		

Figura 12. Comparación de las curvas ROC y la curva de aprendizaje con diferentes ajustes de parámetros

Como resultado de probar diferentes ajustes de parámetros observamos que aumentar el número de estimadores y aumentar el learning rate fueron los ajustes que

mejoran en mejor medida el modelo sin causar overfitting. Probaremos a combinar estos dos ajustes para mejorar el modelo y decidir por uno como el mejor y más robusto modelo. Con learning rate=0.2 y número de estimadores=200 obtenemos los siguientes resultados:

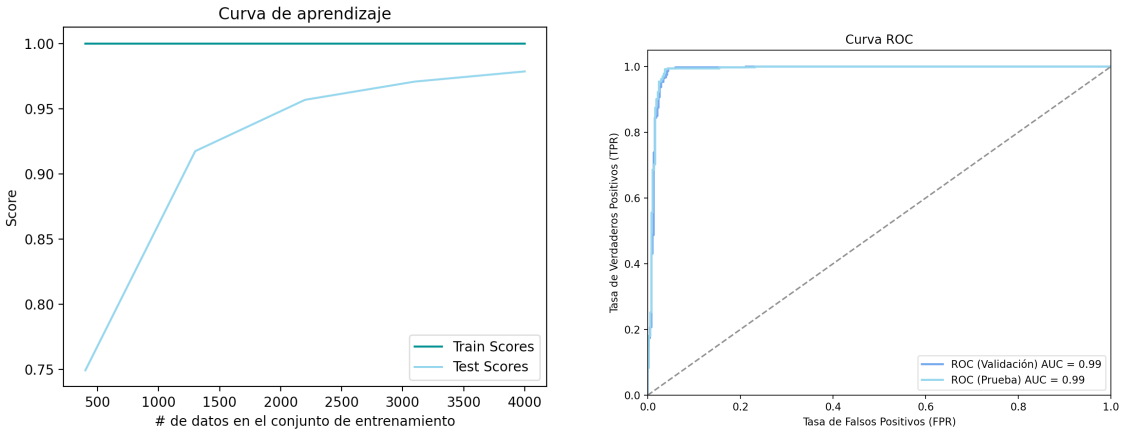


Figura 13. Curva de aprendizaje y curvas ROC del modelo final

Validación	**Prueba**
Accuracy: 0.9787 Recall: 1.0000 Precision: 0.9582 F1 Score: 0.9787 [367 16] [0 367]	Accuracy: 0.9693 Recall: 1.0000 Precision: 0.9424 F1 Score: 0.9703 [351 23] [0 376]

Figura 14. Métricas de evaluación en los conjuntos de validación y prueba usando el modelo final

Además, tenemos que:

- Perdida: 0.0405
- Sesgo: 0.0333
- Varianza: 0.0168

Observamos que con estos parámetros obtenemos un muy buen modelo, al que si le damos muy pocos datos de entrenamiento, como es intuitivo, tiene un desempeño terrible en el conjunto de validación y tiene mucho overfitting, sin embargo, conforme aumentamos la cantidad de datos en el conjunto de entrenamiento mejoramos el puntaje del modelo en el conjunto de validación y parece que captura casi a la perfección los datos sin aprenderlos de memoria. Además, la curva ROC se acerca a demostrar el clasificador perfecto, llegando cerca de (0,1) y el puntaje AUC es igual en ambos conjuntos, lo que sigue demostrando que el modelo es bueno. Al usar las diferentes métricas en los conjuntos de validación y prueba no vemos diferencias significativas e incluso vemos que no existe ningún falso negativo, lo que es ideal para este tipo de problemas, ya que no dejaremos sin tratar a pacientes con riesgo de sufrir un derrame cerebral.

VIII. Conclusión

En conclusión, nuestro primer modelo tenía un buen ajuste pero cierto margen de mejorar, se podría decir que tenía un ligero overfitting al tener mejor ajuste en el conjunto de validación que en el conjunto de prueba, además aunque el modelo era mejor que un clasificador aleatorio (indicado por las métricas y las curvas ROC), tenía posibilidades de un mejor ajuste.

Al explorar la respuesta del modelo ante las modificaciones de parámetros encontramos que parámetros del modelo podíamos ajustar para mejorar el modelo, en este caso modificar el learning rate y el número de estimadores al alza mejora el modelo sin causar un problema grave de overfitting siempre y cuando se tengan las suficientes muestras en el conjunto de entrenamiento. Mejoró sustancialmente el modelo, mejorando las métricas, eliminando los falsos negativos y disminuyendo su sesgo y varianza significativamente, todos los indicadores nos muestran un modelo robusto y bien ajustado.