Instituto Tecnológico de Estudios Superiores de Monterrey

Campus Estado de México

Inteligencia artificial avanzada para la ciencia de datos I (Gpo 101)

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

A01753729 | Marco Antonio Caudillo Morales

Septiembre 11, 2024

Introducción

En el contexto financiero moderno, la aprobación de préstamos es un proceso crucial que requiere la evaluación cuidadosa de múltiples factores para minimizar riesgos. Para optimizar este proceso, este proyecto propone un sistema de predicción de aprobación de préstamos utilizando técnicas de aprendizaje automático, con un enfoque en el uso del modelo de Gradient Boosting, conocido por su capacidad para combinar múltiples modelos débiles en un clasificador robusto y preciso.

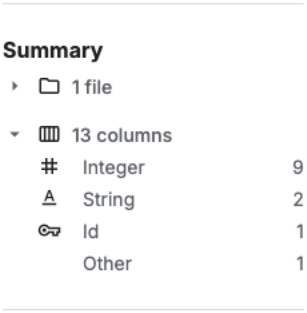
En este documento se encontrará el análisis completo de porque se escogio este modelo junto con todos los análisis de las gráficas.

Como se comento en la introducción, loq ue se busca aumentar la aprobación de préstamos a las personas que cumplan con el perfil y para esto se debe hacer una evaluación cuidadosa.

El dataset que se ocupo es:

https://www.kaggle.com/datasets/architsharma01/loan-approval-prediction-dataset

El conjunto de datos de aprobación de préstamos es una colección de registros financieros e información asociada que se utiliza para determinar la elegibilidad de personas u organizaciones para obtener préstamos de una institución crediticia. Incluye varios factores como la puntuación CIBIL, los ingresos, la situación laboral, el plazo del préstamo, el monto del préstamo, el valor de los activos y el estado del préstamo.



loan_id	no_of_dependents Number of Dependents of the Applicant	education Education of the Applicant	self_employed Employment Status of the Applicant	income_annum Annual Income of the Applicant	loan_amount Loan Amount	loan_term Loan Term in Years	cibil_score Credit Score	residential_asset... residential_assets_value	commercial_ass... commercial_assets_value	luxury_assets_va... luxury_assets_value	bank_asset_value bank_asset_value	loan_status Loan Approval Status
1 4269	0 5	Graduate 50% Not Graduate 50%	true 0 0% false 0 0%	200k 9.90m	300k 39.5m	2 20	300 900	-100000 29.1m	0 19.4m	300k 39.2m	0 14.7m	Approved 62% Rejected 38%
1	2	Graduate	No	9600000	29900000	12	778	2400000	17600000	22700000	8000000	Approved
2	0	Not Graduate	Yes	4100000	12200000	8	417	2700000	2200000	8800000	3300000	Rejected
3	3	Graduate	No	9100000	29700000	20	506	7100000	4500000	33300000	12800000	Rejected
4	3	Graduate	No	8200000	30700000	8	467	18200000	3300000	23300000	7900000	Rejected
5	5	Not Graduate	Yes	9800000	24200000	20	382	12400000	8200000	29400000	5000000	Rejected
6	0	Graduate	Yes	4800000	13500000	10	319	6800000	8300000	13700000	5100000	Rejected
7	5	Graduate	No	8700000	33000000	4	678	22500000	14800000	29200000	4300000	Approved
8	2	Graduate	Yes	5700000	15000000	20	382	13200000	5700000	11800000	6000000	Rejected
9	0	Graduate	Yes	800000	2200000	20	782	1300000	800000	2800000	600000	Approved
10	5	Not Graduate	No	1100000	4300000	10	388	3200000	1400000	3300000	1600000	Rejected
11	4	Graduate	Yes	2900000	11200000	2	547	8100000	4700000	9500000	3100000	Approved
12	2	Not Graduate	Yes	6700000	22700000	18	538	15300000	5800000	20400000	6400000	Rejected
13	3	Not Graduate	Yes	5000000	11600000	16	311	6400000	9600000	14600000	4300000	Rejected
14	2	Graduate	Yes	9100000	31500000	14	679	10800000	16500000	28900000	5000000	Approved
15	1	Not Graduate	No	1900000	7400000	6	469	1900000	1200000	5900000	1900000	Rejected
16	5	Not Graduate	No	4700000	10700000	10	794	5700000	3900000	16400000	4400000	Approved

Como primer paso se hizo la revisión de la base de datos y se observaron las columnas que contenian y se vio si es que no existián valores nulos.

```
loan_id                     0
no_of_dependents            0
education                   0
self_employed               0
income_annum                0
loan_amount                 0
loan_term                   0
cibil_score                 0
residential_assets_value    0
commercial_assets_value     0
luxury_assets_value         0
bank_asset_value            0
loan_status                 0
dtype: int64
```

```
loan_status
Approved    12569
Rejected     7431
Name: count, dtype: int64
```

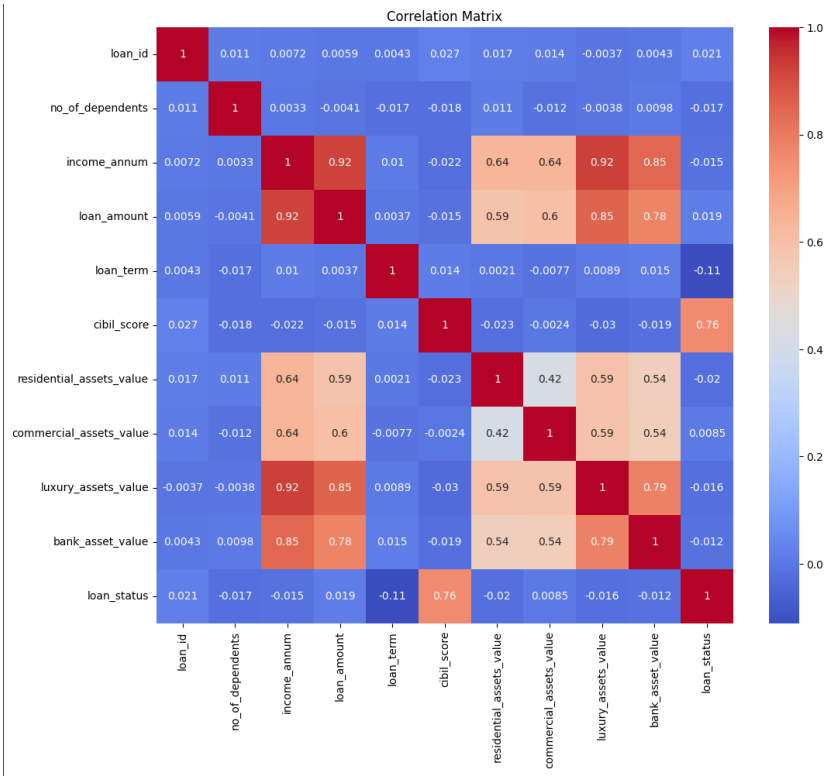
Una vez observado y al no tener valores nulos lo que se busco fue que “loan_status” que es nuestra variable objetivo fuera binario por lo que se realizó el cambio.

$$\{'Approved': 1, 'Rejected': 0\}$$

	loan_id	no_of_dependents	education	self_employed	income_annum	loan_amount	loan_term	cibil_score	residential_assets_value	commercial_assets_value	luxury_assets_value	bank_asset_value	loan_status
0	1	2	Graduate	No	9600000	29900000	12	778	2400000	17600000	22700000	8000000	1
1	2	0	Not Graduate	Yes	4100000	12200000	8	417	2700000	2200000	8800000	3300000	0
2	3	3	Graduate	No	9100000	29700000	20	506	7100000	4500000	33300000	12800000	0
3	4	3	Graduate	No	8200000	30700000	8	467	18200000	3300000	23300000	7900000	0
4	5	5	Not Graduate	Yes	9800000	24200000	20	382	12400000	8200000	29400000	5000000	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...
19995	1277	5	Graduate	Yes	2894457	11869799	14	475	6093737	5159712	12391821	4076310	0
19996	886	2	Graduate	No	5537613	13187189	1	730	7332788	5351765	15758604	6258350	1
19997	1050	0	Graduate	Yes	1088743	3913150	7	537	2449059	977768	2229851	1288266	1
19998	3478	4	Graduate	No	9893177	37845942	3	732	13103524	11225347	34655659	14008810	1
19999	1771	4	Not Graduate	Yes	3593232	10486490	9	338	9680447	5882830	10015299	4314935	0

20000 rows × 13 columns

Una vez con los cambios previos lo que se busco realizar matriz de correlación para analizar nuestras variables que eran altamente dependientes con nuestra variable objetivo.



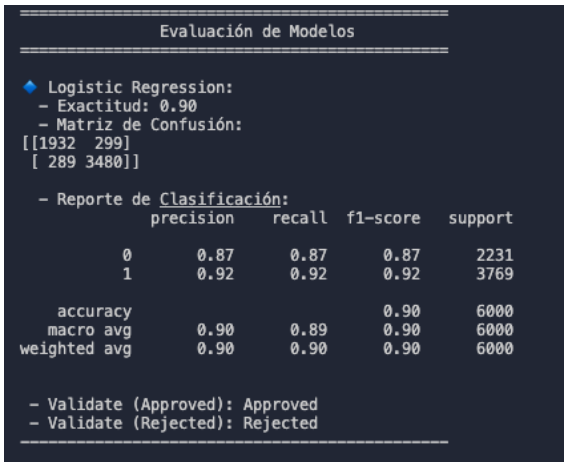
Las variables dependientes que se escogieron fue las de:

'cibil_score', 'loan_term', 'no_of_dependents'

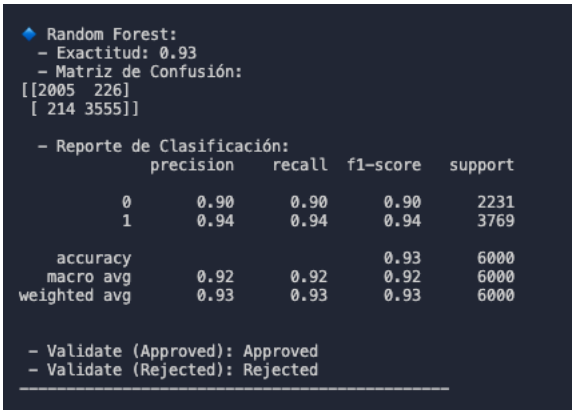
Para este punto se ocupo la librería de “sklearn” para poder dividir el dataset en train y test (70% entrenamiento y 30% de prueba)

```
# Separar los datos en conjuntos de entrenamiento y prueba (70% entrenamiento, 30% prueba)
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.3, random_state=42)
```

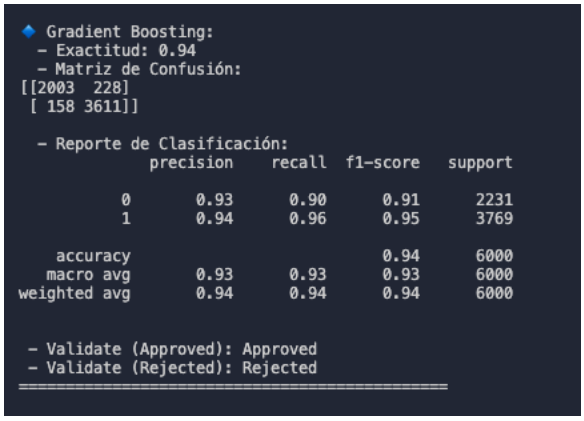
Con esto ya pudimos ocupar diferentes modelos para poder ver cuál es el que mejor resultados nos daban.



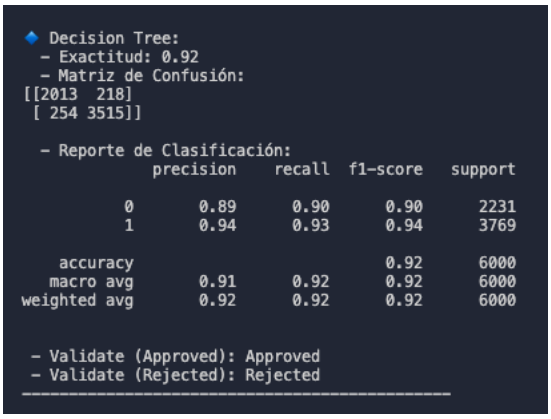
```
==========================================
            Evaluación de Modelos
==========================================

🔷 Logistic Regression:
  - Exactitud: 0.90
  - Matriz de Confusión:
[[1932  299]
 [ 289 3480]]

  - Reporte de Clasificación:
              precision    recall  f1-score   support

           0       0.87      0.87      0.87      2231
           1       0.92      0.92      0.92      3769

    accuracy                           0.90      6000
   macro avg       0.90      0.89      0.90      6000
weighted avg       0.90      0.90      0.90      6000


 - Validate (Approved): Approved
 - Validate (Rejected): Rejected
------------------------------------------
```



```
🔷 Random Forest:
  - Exactitud: 0.93
  - Matriz de Confusión:
[[2005  226]
 [ 214 3555]]

  - Reporte de Clasificación:
              precision    recall  f1-score   support

           0       0.90      0.90      0.90      2231
           1       0.94      0.94      0.94      3769

    accuracy                           0.93      6000
   macro avg       0.92      0.92      0.92      6000
weighted avg       0.93      0.93      0.93      6000


 - Validate (Approved): Approved
 - Validate (Rejected): Rejected
------------------------------------------
```



```
🔷 Gradient Boosting:
  - Exactitud: 0.94
  - Matriz de Confusión:
[[2003  228]
 [ 158 3611]]

  - Reporte de Clasificación:
              precision    recall  f1-score   support

           0       0.93      0.90      0.91      2231
           1       0.94      0.96      0.95      3769

    accuracy                           0.94      6000
   macro avg       0.93      0.93      0.93      6000
weighted avg       0.94      0.94      0.94      6000


 - Validate (Approved): Approved
 - Validate (Rejected): Rejected
==========================================
```



```
🔷 Decision Tree:
  - Exactitud: 0.92
  - Matriz de Confusión:
[[2013  218]
 [ 254 3515]]

  - Reporte de Clasificación:
              precision    recall  f1-score   support

           0       0.89      0.90      0.90      2231
           1       0.94      0.93      0.94      3769

    accuracy                           0.92      6000
   macro avg       0.91      0.92      0.92      6000
weighted avg       0.92      0.92      0.92      6000


 - Validate (Approved): Approved
 - Validate (Rejected): Rejected
------------------------------------------
```

El modelo que se ha escogido tras el análisis es **Gradient Boosting** por las siguientes razones:

1. Exactitud:

- **Gradient Boosting** tiene la mayor exactitud con un 0.94, lo que indica que predice correctamente el 94% de las veces.
- Otros modelos como el **Decision Tree** (0.92), **Random Forest** (0.93), y **Logistic Regression** (0.90) tienen una exactitud ligeramente inferior.

2. Matriz de Confusión:

- La matriz de confusión del modelo de **Gradient Boosting** muestra:
  - Falsos negativos y falsos positivos más bajos en comparación con otros modelos.
  - En la clase "0" (Rejected), tiene 2003 verdaderos negativos (correctamente rechazados) y solo 228 falsos positivos (mal clasificados como aprobados).
  - En la clase "1" (Approved), tiene 3611 verdaderos positivos (correctamente aprobados) y solo 158 falsos negativos (mal clasificados como rechazados).
- Comparando con otros modelos, **Gradient Boosting** tiene menos errores (falsos positivos y falsos negativos).

3. Reporte de Clasificación:

- **Gradient Boosting** tiene los mejores valores de:
  - **Recall** (Sensibilidad): 0.96 para la clase "1" (Approved), lo que indica que el modelo es capaz de identificar correctamente el 96% de las solicitudes aprobadas, lo cual es importante cuando se quiere minimizar el riesgo de no aprobar una buena solicitud.
  - **F1-Score**: Con 0.95 para la clase "1", muestra que tiene un buen balance entre precisión y recall.
  - Otros modelos como **Decision Tree** y **Random Forest** también tienen buenos F1-scores, pero son inferiores en comparación a Gradient Boosting (0.94 frente a 0.95 en la clase "1").

4. Precisión y Recall en Clase Rechazada (0):

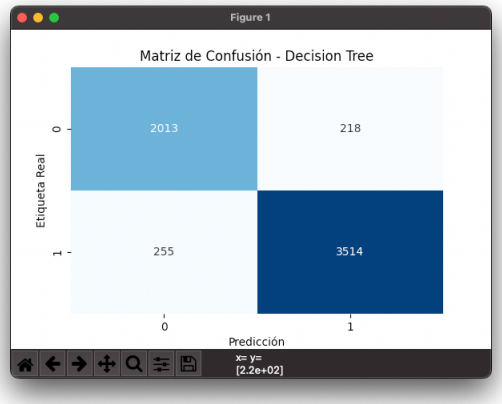
- **Gradient Boosting** tiene una precisión de 0.93 para la clase "0" (Rejected), lo que significa que es preciso en predecir cuando una solicitud debe ser rechazada.

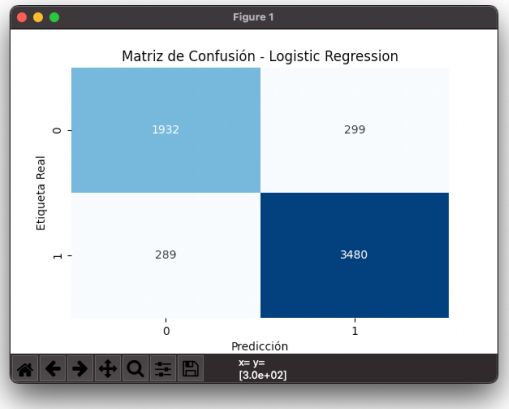
- Otros modelos como el **Decision Tree** y **Random Forest** también tienen buenos resultados en esta métrica, pero su recall (la capacidad de identificar correctamente todos los rechazos) es ligeramente inferior a Gradient Boosting.

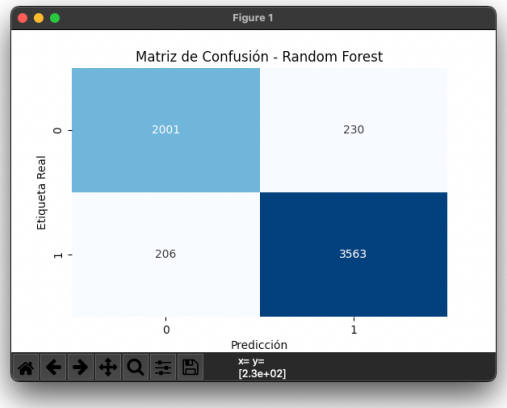
Conclusión:

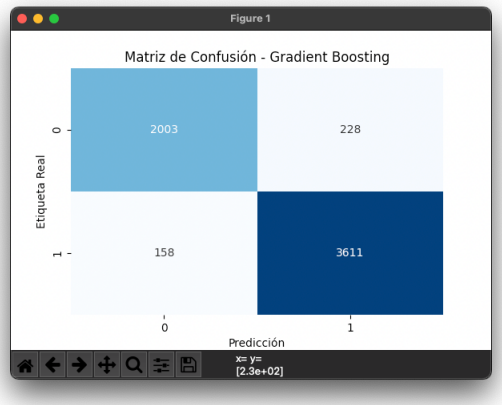
Gradient Boosting se destaca porque no solo tiene la mejor exactitud general (0.94), sino que también muestra una excelente capacidad de balancear tanto la precisión como el recall en ambas clases (aprobada y rechazada), reduciendo significativamente tanto los falsos positivos como los falsos negativos. Esto es particularmente importante en contextos donde los errores en la clasificación de aprobaciones y rechazos de préstamos pueden tener consecuencias costosas.

Por lo tanto, **Gradient Boosting** es el mejor modelo en este caso debido a su capacidad de generar predicciones más equilibradas y su menor tasa de errores en comparación con los otros modelos.

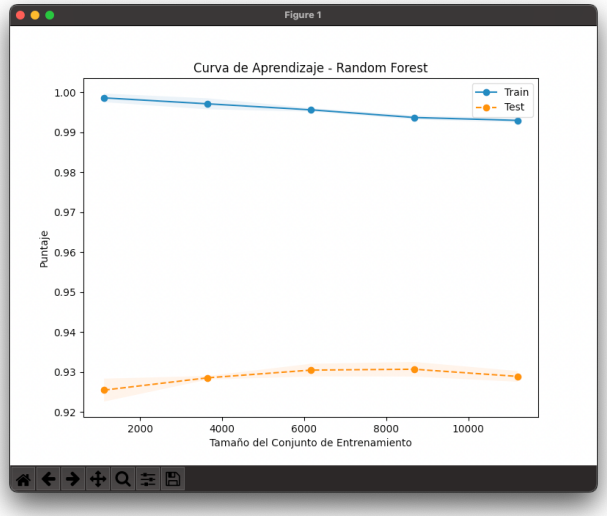


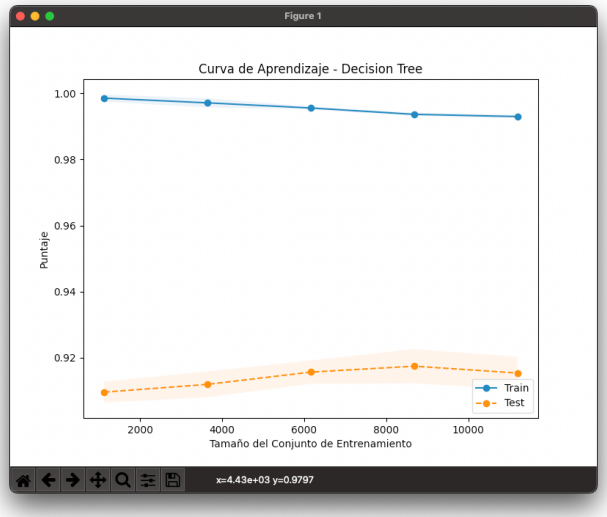


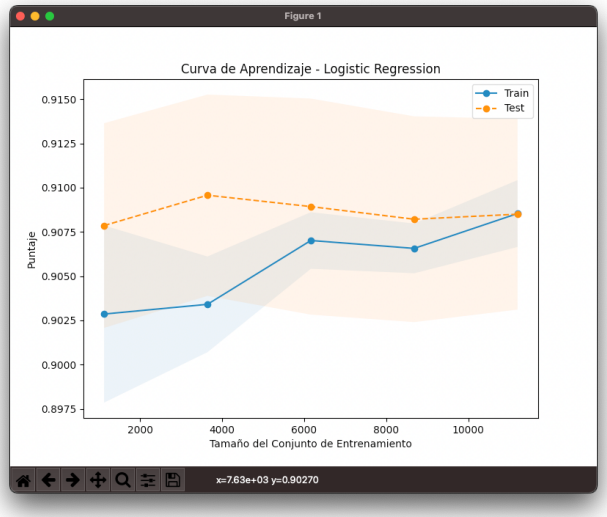




Para hacer el análisi de Overfitting se ocuparon las gráficas de Curvas de Aprendizaje







Árbol de Decisión

Interpretación:

Curva de Entrenamiento:

La línea azul muestra que el modelo de árbol de decisión tiene una precisión muy alta (~100%) en el conjunto de entrenamiento, lo que indica que está ajustando perfectamente los datos de entrenamiento.

Curva de Prueba:

La línea naranja muestra que la precisión en el conjunto de prueba es considerablemente menor (~92%).

Conclusión:

Esto indica un claro sobreajuste (overfitting). El modelo se adapta demasiado a los datos de entrenamiento y no generaliza bien en nuevos datos. Para mejorar esto, sería recomendable podar el árbol o ajustar algunos hiperparámetros como la profundidad máxima del árbol.

Gradient Boosting

Interpretación:

Curva de Entrenamiento:

La precisión en el conjunto de entrenamiento es alta (~98%), pero no alcanza el 100%, lo que indica que el modelo no está completamente sobreajustado.

Curva de Prueba:

La precisión en el conjunto de prueba es similar a la del entrenamiento (~92-93%), lo que indica una buena capacidad de generalización.

Conclusión:

El modelo de Gradient Boosting parece estar funcionando bien, logrando un buen equilibrio entre ajuste y generalización. Es menos probable que esté sobreajustado en comparación con el modelo de árbol de decisión.

Regresión Logística

Interpretación:

Curva de Entrenamiento:

La precisión en el conjunto de entrenamiento mejora gradualmente, pero nunca alcanza valores muy altos, quedándose alrededor del 90.25%.

Curva de Prueba:

La precisión en el conjunto de prueba es también alrededor del 90.75%.

Conclusión:

El modelo de Regresión Logística muestra un rendimiento bastante consistente entre los conjuntos de entrenamiento y prueba, lo que indica que el modelo está generalizando bien y no está sobreajustado. Es un buen modelo de base, aunque podría no captar las relaciones más complejas en los datos.

Random Forest

Interpretación:

Curva de Entrenamiento:

La precisión del conjunto de entrenamiento es muy alta (~99-100%).

Curva de Prueba:

La precisión en el conjunto de prueba es significativamente más baja (~92%), pero se mantiene estable.

Conclusión:

Al igual que el árbol de decisión, Random Forest también muestra un cierto grado de sobreajuste, aunque no tan extremo. Random Forest tiende a mejorar la capacidad de generalización en comparación con los árboles de decisión individuales, pero todavía hay margen de mejora en la diferencia entre las curvas de entrenamiento y prueba.

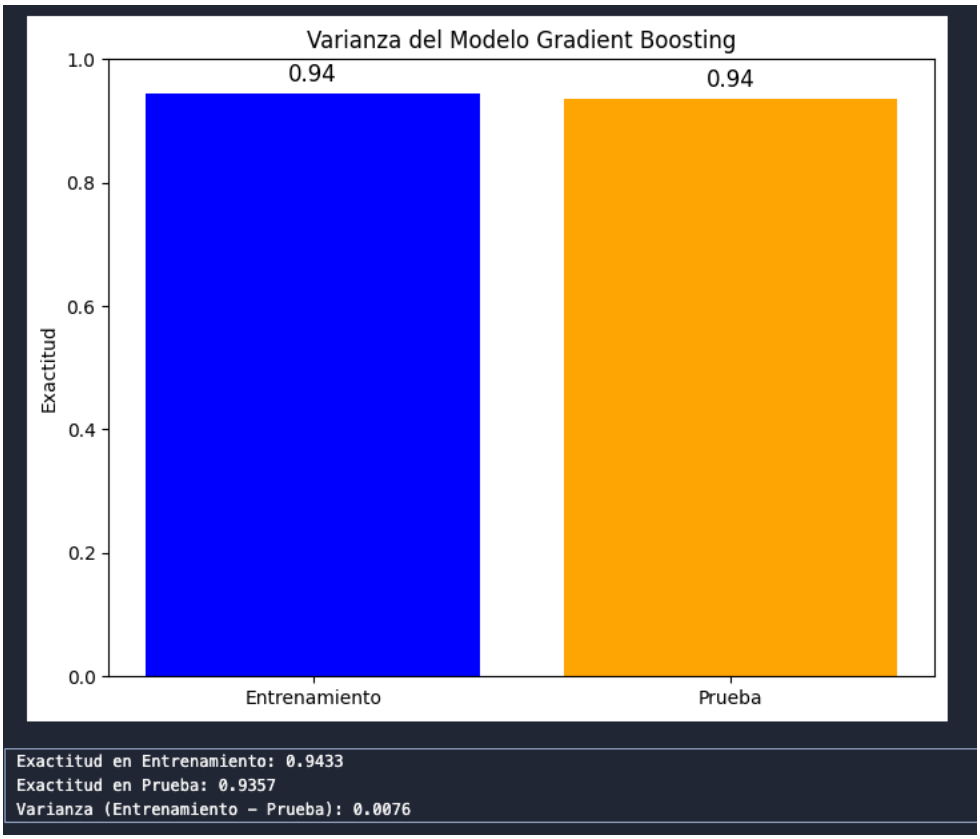
Conclusión General

Regresión Logística: Buen rendimiento con poca o ninguna señal de sobreajuste. Funciona bien en problemas lineales.
Árbol de Decisión: El modelo está claramente sobreajustado y necesita una mayor regularización o poda.
Random Forest: Mejora en la capacidad de generalización respecto a los árboles de decisión, pero aún muestra signos de sobreajuste.
Gradient Boosting: Presenta el mejor equilibrio entre ajuste y generalización, siendo el modelo más efectivo en términos de rendimiento.

Recomendaciones
Regresión Logística es una opción segura y simple cuando las relaciones entre las variables son lineales.
Gradient Boosting es probablemente la mejor opción cuando se desea un modelo que generalice bien y capture relaciones no lineales entre las variables.
Si se usa **Árbol de Decisión** o **Random Forest**, es recomendable ajustar sus hiperparámetros para reducir el sobreajuste, como la profundidad máxima o el número mínimo de muestras en las hojas.

```
# Datos de prueba para validar los modelos, con un buen y un mal historial crediticio
# Buen historial crediticio
cibil_score_good = 778
loan_term_good = 12
dependents_good = 2

# Escalar los datos
input_data_good = scaler.transform(
    [[cibil_score_good, loan_term_good, dependents_good]])

# Mal historial crediticio
cibil_score_bad = 417
loan_term_bad = 8
dependents_bad = 0

# Escalar los datos
input_data_bad = scaler.transform(
    [[cibil_score_bad, loan_term_bad, dependents_bad]])
```

Dentro del código vienen datos de prueba para validar los modelos, estos se encuentran en las imagenes de evaluación.

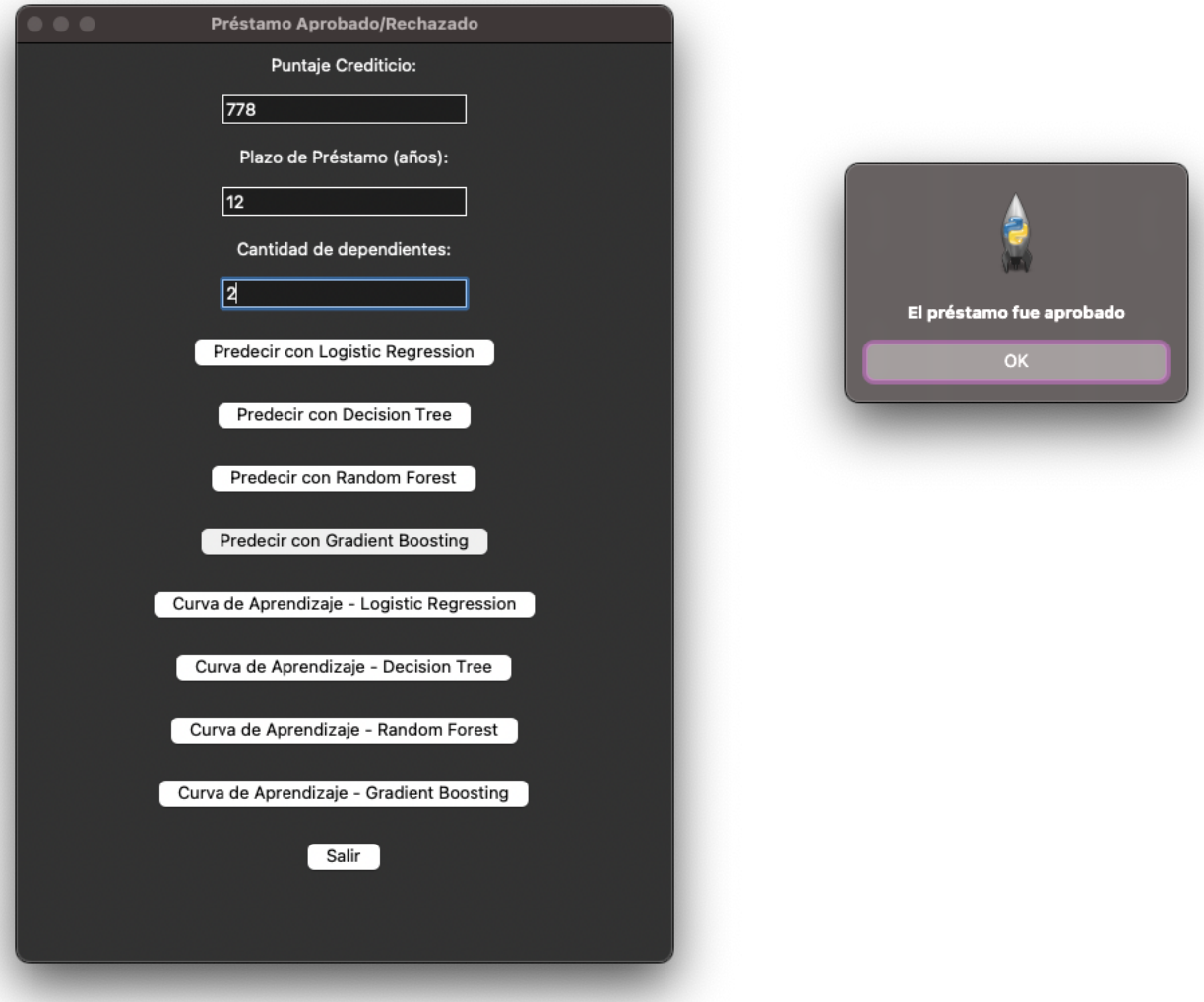


Interpretación:

1. **Baja Varianza**: Una varianza de solo **0.0076** indica que el modelo tiene un buen equilibrio entre aprender de los datos de entrenamiento y generalizar bien en los datos de prueba. El modelo no ha memorizado los datos de entrenamiento ni se ha sobreajustado, lo que es un resultado excelente para un problema de clasificación como este.
2. **Exactitud Alta en Ambos Conjuntos**: Tanto la exactitud en entrenamiento (94.33%) como la exactitud en prueba (93.57%) son muy altas, lo que sugiere que el modelo está aprendiendo patrones relevantes de los datos sin caer en un sobreajuste. Esto es ideal en situaciones donde la exactitud es crítica, como en la predicción de aprobaciones de préstamos.

El modelo de **Gradient Boosting** ha demostrado ser robusto, con un bajo nivel de varianza, lo que indica que no está sobreajustado a los datos de entrenamiento y generaliza bien en los datos de prueba. Este resultado sugiere que los ajustes realizados en los hiperparámetros del modelo han sido efectivos, logrando un equilibrio óptimo entre el ajuste del modelo y su capacidad de generalización.

Para este proyecto se opto por una interfaz gráfica para poder interactuar con el modelo.



Las instrucciones para instalar y correr el proyecto se encuentra en el archivo README del repositorio.

Se intentarón realizar estos cambios, pero el tiempo de ejecución es demasiado alto.

Ajuste de Hiperparámetros:

- **Número de estimadores (n_estimators)**: Controlar cuántos árboles de decisión se entrenan en el modelo de boosting. Incrementar este valor puede ayudar a mejorar el rendimiento hasta cierto punto, aunque si es demasiado alto, puede causar sobreajuste.
- **Profundidad máxima de los árboles (max_depth)**: Controla la profundidad máxima que puede alcanzar cada árbol en el modelo de boosting. Si es muy profundo, el modelo puede sobreajustarse a los datos de entrenamiento. Reducir este valor ayuda a que los árboles sean más simples y generalicen mejor.
- **Tasa de aprendizaje (learning_rate)**: Esta es la contribución de cada árbol al modelo final. Si la tasa de aprendizaje es demasiado alta, el modelo puede aprender demasiado rápido y sobreajustarse. Una tasa de aprendizaje más baja permite un ajuste más gradual.
- **Fracción mínima de datos en las hojas (min_samples_leaf)**: Esta métrica controla cuántos datos debe haber en una hoja de decisión. Si es demasiado bajo, el modelo puede ser muy sensible al ruido. Ajustarlo hacia arriba puede mejorar la robustez.

Control de la tasa de aprendizaje (learning_rate):

- Reducir la tasa de aprendizaje obliga al modelo a aprender de forma más lenta y gradual, lo que ayuda a evitar el sobreajuste. Una tasa de aprendizaje más baja, junto con un mayor número de árboles, permite que el modelo construya de manera incremental una predicción más precisa y robusta.

Subsampleo (subsample):

- Usar solo una fracción de los datos en cada iteración introduce una especie de aleatoriedad que reduce la varianza y previene el sobreajuste. Este enfoque es similar al que utiliza el **Random Forest**, pero en el contexto del **Gradient Boosting**, ayuda a que el modelo sea más robusto al ruido de los datos.

Regularización:

- La regularización L1 y L2 ayuda a reducir la complejidad del modelo, evitando que los coeficientes se vuelvan demasiado grandes, lo que a su vez evita el sobreajuste y mejora la generalización del modelo.

Después de aplicar estas técnicas de ajuste de hiperparámetros y regularización, deberías observar que el modelo de **Gradient Boosting** mejora en términos de:

- **Exactitud** en los datos de prueba.
- **Reducción del sobreajuste**, es decir, una menor discrepancia entre el rendimiento en los datos de entrenamiento y los datos de prueba.
- **Mejora de las métricas de precisión, recall y F1-score**, particularmente en las clases minoritarias (como la clase "rechazada").

Conclusión

En este proyecto, implementamos y evaluamos múltiples modelos de clasificación para predecir la aprobación o rechazo de préstamos, utilizando un conjunto de datos relacionados con el historial crediticio de los solicitantes. Los modelos que se implementaron incluyen **Regresión Logística**, **Árboles de Decisión**, **Bosques Aleatorios** y **Gradient Boosting**. Después de analizar el rendimiento de cada modelo a través de métricas como la exactitud, la matriz de confusión, y el reporte de clasificación (precisión, recall, F1-score), identificamos que el modelo de **Gradient Boosting** tuvo el mejor desempeño general.