ENTREGA FINAL DEL PROYECTO

Por:

Jonatan Stiven Restrepo Lora

CC:1018376574

Andres Felipe Graciano Monsalve

CC: 71375739

Ricardo Osorio Castro

CC: 1036785264

Materia:

Introducción a la Inteligencia Artificial

Métodos Estadísticos

Profesor

Raul Ramos Pollan

Facultad de Ingeniería

Universidad de Antioquia - Colombia

2023

CONTENIDO

1. INTRODUCCIÓN

¿Cómo saben los emisores de tarjetas que devolveremos lo que cobramos? La predicción del incumplimiento crediticio es fundamental para gestionar el riesgo en un negocio de préstamos al consumo. La predicción del incumplimiento crediticio permite a los prestamistas optimizar las decisiones crediticias, lo que conduce a una mejor experiencia del cliente y a una economía empresarial sólida.

El objetivo del modelo es predecir la probabilidad de que un cliente no pague el saldo de su tarjeta de crédito en el futuro en función de su perfil de cliente mensual. La variable binaria objetivo se calcula observando una ventana de rendimiento de 18 meses después del último extracto de la tarjeta de crédito, y si el cliente no paga el monto adeudado en 120 días después de la última fecha del extracto, se considera un evento de incumplimiento.

Vamos a usar el dataset de kaggle de esta competición (https://www.kaggle.com/competitions/amex-default-prediction/data?select=train_data.csv),El objetivo de esta competencia es predecir la probabilidad de que un cliente no pague el saldo de su tarjeta de crédito en el futuro en función de su perfil de cliente mensual. La variable binaria objetivo se calcula observando una ventana de rendimiento de 18 meses después del último extracto de la tarjeta de crédito, y si el cliente no paga el monto adeudado en 120 días después de la última fecha del extracto, se considera un evento de incumplimiento.El conjunto de datos contiene características de perfil agregadas para cada cliente en cada fecha de estado de cuenta. Las funciones son anónimas y normalizadas y se clasifican en las siguientes categorías generales:

D_* = Delinquency variables (Variables de morosidad)
S_* = Spend variables (Variables de Gasto)
P_* = Payment variables (Variables de Pago)
B_* = Balance variables (Variables de Balance)
R_* = Risk variables (Variables de Riesgo)

La tarea es predecir para cada customerID la probabilidad de un incumplimiento de pago futuro (target=1). Tenga en cuenta que la clase negativa se ha submuestreo para este conjunto de datos al 5 % y, por lo tanto, recibe una ponderación de 20 veces en la métrica de puntuación.

El desempeño deseable en producción para el modelo de predicción de incumplimiento en el pago de tarjetas de crédito será tener un Recall Alto, dado que el objetivo principal es identificar a los clientes que no pagarán a tiempo (incumplimientos), un recall alto es fundamental. Esto significa que el modelo está capturando la mayoría de los incumplimientos, lo que ayuda a reducir el riesgo de pérdidas financieras para la institución financiera.

2. EXPLORACIÓN DESCRIPTIVA DEL DATASET

Para comenzar con la tarea predictiva, se inició el trabajo con el conjunto de datos suministrados. Sin embargo, estos datos resultaron ser bastante voluminosos. Tiene 924621 número de muestras y 396 columnas de todo el dataset.
En la competencia podemos ver que el tamaño total del dataset es de 50.31GB.

Por lo tanto, se decidió seleccionar aleatoriamente una muestra de aproximadamente 6000 registros para su posterior entrenamiento.
Estos registros se guardaron y se sometieron a una evaluación con el propósito de identificar sus variables categóricas y determinar qué operaciones eran necesarias para su procesamiento. Luego, después de un análisis y filtro, se procedió a llevar a cabo la tarea de limpieza, que incluyó la imputación de valores faltantes de acuerdo a los resultados de los análisis previamente diseñados.

Selección de muestras

Se comenzó por cargar los conjuntos de datos, train_data y train_labels, utilizando la herramienta de pandas. Sin embargo, debido al tamaño de los archivos, esta tarea resultó ser inviable. Como solución, se optó por utilizar Spark, dado que permite manejar volúmenes de datos más grandes.

```
spark = SparkSession.builder.getOrCreate()
```

```
data = spark.read.csv('file:///E:/data/amex-default-prediction/train_data.csv', header=True)
data_label = spark.read.csv('file:///E:/data/amex-default-prediction/train_labels.csv', header=True)
```

Leemos los archivos, el primero contiene toda la información, el segundo los resultados y procedemos a mirar qué columnas tienen en común para hacer una unión por el campo en común.

```
data.columns
```

```
['customer_ID',
 'S_2',
 'P_2',
 'D_39',
 'B_1',
 'B_2',
 'R_1',
 'S_3',
 'D_41',
 'B_3',
```

```
data_label.columns
```

```
['customer_ID', 'target']
```

```
data.select('customer_ID', 'B_30', 'B_38', 'D_114', 'D_116', 'D_117', 'D_120', 'D_126', 'D_63', 'D_64', 'D_66', 'D_68').show()
```

```
+--------------------+----+----+-----+-----+-----+-----+-----+----+----+----+----+
|         customer_ID|B_30|B_38|D_114|D_116|D_117|D_120|D_126|D_63|D_64|D_66|D_68|
+--------------------+----+----+-----+-----+-----+-----+-----+----+----+----+----+
|0000099d6bd597052...| 0.0| 2.0|  1.0|  0.0|  4.0|  0.0|  1.0|  CR|   O|NULL| 6.0|
|0000099d6bd597052...| 0.0| 2.0|  1.0|  0.0|  4.0|  0.0|  1.0|  CR|   O|NULL| 6.0|
|0000099d6bd597052...| 0.0| 2.0|  1.0|  0.0|  4.0|  0.0|  1.0|  CR|   O|NULL| 6.0|
|0000099d6bd597052...| 0.0| 2.0|  1.0|  0.0|  4.0|  0.0|  1.0|  CR|   O|NULL| 6.0|
|0000099d6bd597052...| 0.0| 2.0|  1.0|  0.0|  4.0|  0.0|  1.0|  CR|   O|NULL| 6.0|
|0000099d6bd597052...| 0.0| 2.0|  1.0|  0.0|  4.0|  0.0|  1.0|  CR|   O|NULL| 6.0|
|0000099d6bd597052...| 0.0| 2.0|  1.0|  0.0|  4.0|  0.0|  1.0|  CR|   O|NULL| 6.0|
|0000099d6bd597052...| 0.0| 2.0|  1.0|  0.0|  4.0|  0.0|  1.0|  CR|   O|NULL| 6.0|
|0000099d6bd597052...| 0.0| 2.0|  1.0|  0.0|  4.0|  0.0|  1.0|  CR|   O|NULL| 6.0|
|0000099d6bd597052...| 0.0| 2.0|  1.0|  0.0|  4.0|  0.0|  1.0|  CR|   O|NULL| 6.0|
|0000099d6bd597052...| 0.0| 2.0|  1.0|  0.0|  4.0|  0.0|  1.0|  CR|   O|NULL| 6.0|
|0000099d6bd597052...| 0.0| 2.0|  1.0|  0.0|  4.0|  0.0|  1.0|  CR|   O|NULL| 6.0|
|0000099d6bd597052...| 0.0| 2.0|  1.0|  0.0|  4.0|  0.0|  1.0|  CR|   O|NULL| 6.0|
```

Figura 1: Comparación de datos

Luego, se procedió a identificar las columnas similares con el fin de fusionar los datos y

trabajar con un solo conjunto que incluyera tanto las variables predictoras como el objetivo. Se consideró la variable customer_ID como candidata para este proceso. Sin embargo, al intentar llevar a cabo la operación, se presentaron errores, como el inesperado aumento en el tamaño del conjunto de datos. Esto motivó la necesidad de primero dividir la muestra para poder gestionarla de manera adecuada y proceder a guardarla en un .xslx

```
data_muestra = data.sample(False, 6000 / data.count())
```

```
data_muestra.count()
```

```
6041
```

```
data_muestra.toPandas().to_excel('muestra.xlsx', index=False)
```

```
Guardamos el conjunto de datos en un excel para poder trabajar con el más fácil y poder hacer la unión con los label
```

Figura 2: Guardado de los datos

Merge Datasets

Una vez los datos divididos, procedemos a hacer un merge de la muestra extraída y train_labels para así poder agregar el objetivo correspondiente al customer_ID de nuestro conjunto de datos para proceder a guardarlo dado que este será el dataset con el que estaremos trabajando.

```
print(f'cantidad de registros: {len(pd_label)}')
```

```
cantidad de registros: 458913
```

```
data_full = pd_muestra.merge(pd_label, on='customer_ID')
data_full.head()
```

	customer_ID	S_2	P_2	D_39	B_1	B_2	R_1	S_3	D_41	B_3	...	D_137	D_138
0	000adf2938f771f75a581b65107024eddeae70684778c0...	2017-04-25	0.885999	0.009377	0.008894	1.006219	0.008716	0.065198	0.001650	0.009946	...	NaN	NaN
1	001a152e1893ab8372e7c9627c9de2e024399f2660d5d8...	2017-11-10	0.750890	0.037378	0.045959	1.001278	0.000312	0.087636	0.004212	0.001730	...	NaN	NaN
2	001e2ceaf1421f1477c0de9ba1c9357b9d278f7b670ab7...	2018-03-26	0.607227	0.002678	0.001271	0.812964	0.005415	NaN	0.005049	0.006548	...	NaN	NaN
3	0034f7e366a41d2500643c7dd0faa6302ce944743ccdf5...	2018-01-07	0.775546	0.003486	0.234352	0.040793	0.256024	0.172966	0.000081	0.225639	...	NaN	NaN
4	00394e07aa3f71174f8bedfd16d64f194c80ad9445e17f...	2017-04-27	0.358659	0.001814	0.028545	1.004951	0.009079	0.765318	0.003546	0.009843	...	NaN	NaN

5 rows × 191 columns

```
print(f'cantidad de registros: {len(data_full)}')
```

```
cantidad de registros: 6041
```

Una vez comprobado que el merge lo altero el DF y se procede a guardarlos en un csv y txt

```
data_full.to_excel('data_full.xlsx', index=False)
```

```
data_full.to_csv('data_full.txt', sep='\t', index=False)
```

Figura 3: Muestra completa

Datos categóricos

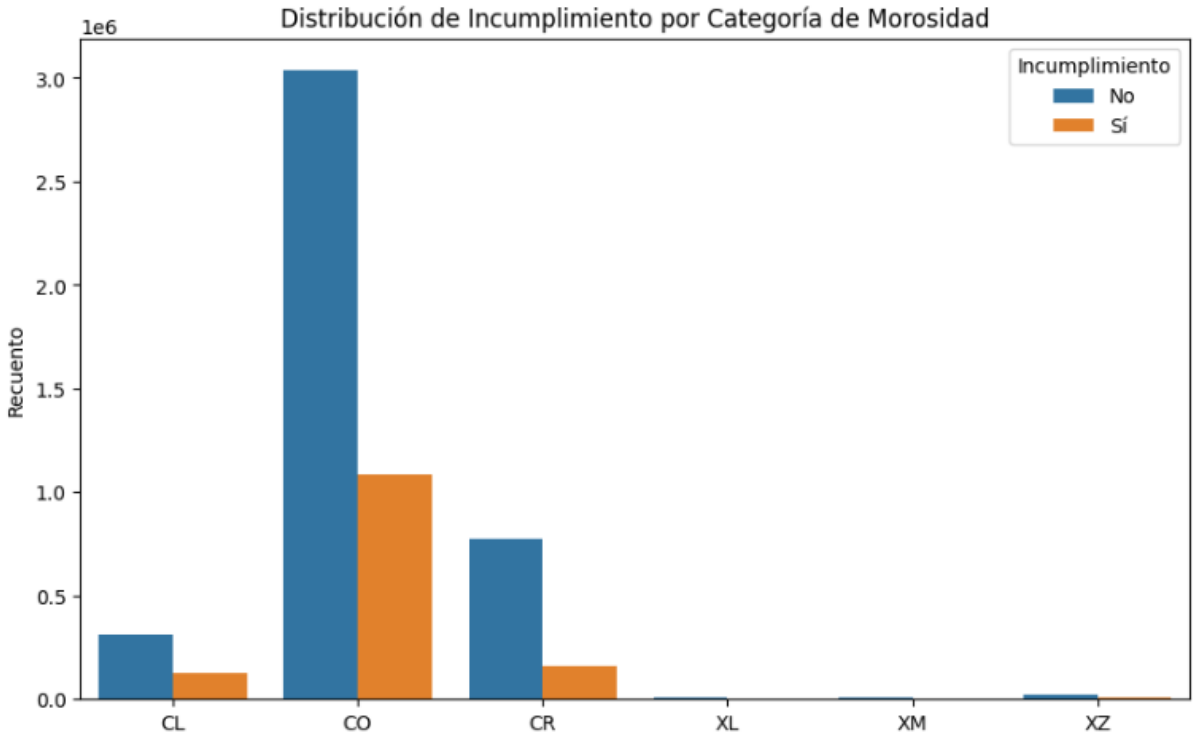
Para las siguientes variables pudimos observar que eran categóricas, con los siguientes valores:

```
D_63 ['CL' 'CO' 'CR' 'XL' 'XM' 'XZ']
D_64 ['-1' 'O' 'R' 'U']
D_66 [0. 1.]
D_68 [0. 1. 2. 3. 4. 5. 6.]
B_30 [0. 1. 2.]
B_38 [1. 2. 3. 4. 5. 6. 7.]
D_114 [0. 1.]
D_116 [0. 1.]
D_117 [-1.  1.  2.  3.  4.  5.  6.]
D_120 [0. 1.]
D_126 [-1.  0.  1.]
```

Podemos observar que la variable D_63 contiene unos valores un tanto extraños, pues aqui esta su posible definición:

A menudo, estos códigos se utilizan para indicar diferentes estados de morosidad o incumplimiento de un préstamo o deuda. Algunos ejemplos comunes de significados en el contexto financiero podrían ser:
-'CL': Collection (en inglés), que indica que la cuenta se encuentra en proceso de cobro.
-'CO': Charged Off (en inglés), que significa que la deuda se considera incobrable.
-'CR': Current (en inglés), que indica que la cuenta está al día y no hay morosidad.
-'XL', 'XM', 'XZ': Estos códigos pueden ser específicos de un sistema de codificación particular y no es posible determinar su significado sin más información.

Con estos datos realizamos un gráfico para ver la relación entre estas y la variable target:



Cómo observamos que las tres últimas columnas apenas y aparecen en el dataset las unimos en una y pasamos a hacer la codificación one-hot.

3. ITERACIONES DE DESARROLLO

3.1. Preprocesado de datos

Limpieza de Datos

En el proceso de limpieza de datos, comenzamos por identificar las variables categóricas, a pesar de que ya se habían especificado en el conjunto de datos original como ’B_30’, ’B_38’, ’D_114’, ’D_116’, ’D_117’, ’D_120’, ’D_126’, ’D_63’, ’D_64’, ’D_66’, ’D_68’.

Estas variables se dividieron en dos categorías: nominales y ordinales. En primer lugar, se llevó a cabo la limpieza y transformación de las variables nominales.
Dentro de las variables nominales, se identificaron ’S_2’, ’D_63’ ’D_64’. Se observó que ’S_2’ representaba fechas, por lo que se procedió a ajustar el tipo de dato utilizando la biblioteca pandas. Para las demás variables nominales, se completaron los valores faltantes utilizando la moda de los datos.
Posteriormente, se aplicó el método de codificación One-Hot Encoding para organizar estas variables categóricas.

```
df = pd.get_dummies(df, columns=['D_63', 'D_64'])
```

```
df.rename(columns={'D_64_-1': 'D_64_I'}, inplace=True)
```

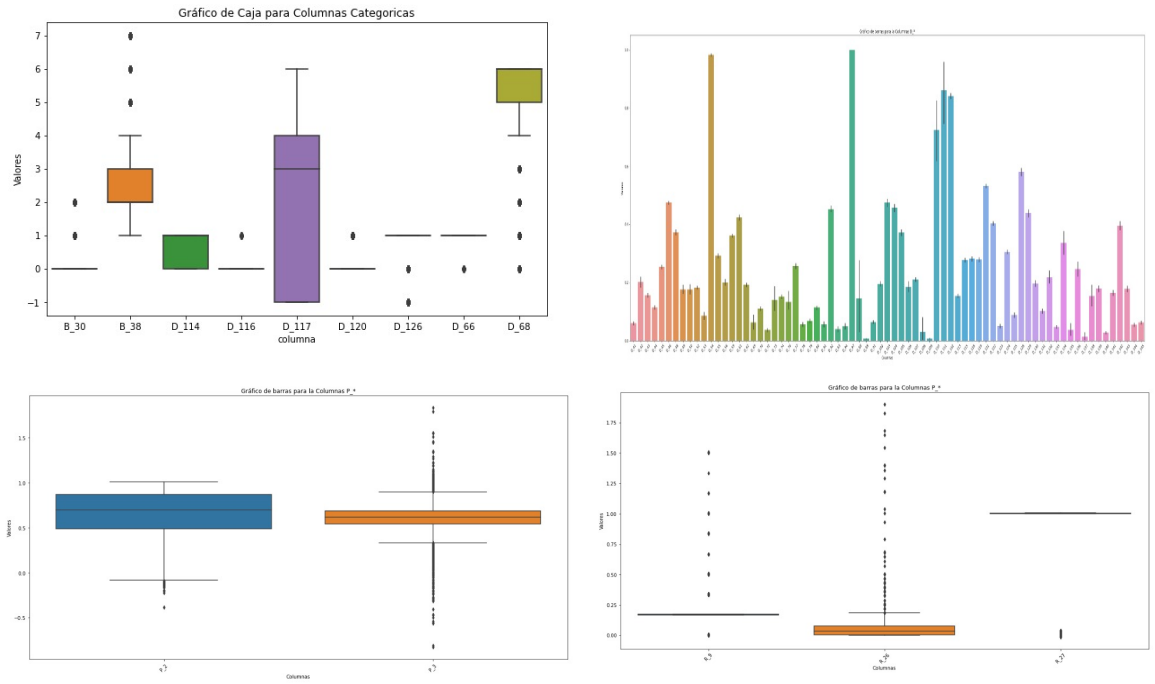
```
df[['D_63_CO', 'D_63_CR', 'D_63_CL', 'D_63_XZ', 'D_63_XZ', 'D_63_XM', 'D_63_XL', 'D_64_O', 'D_64_U', 'D_64_R',
    'D_64_I', 'target']]
```

	D_63_CO	D_63_CR	D_63_CL	D_63_XZ	D_63_XZ	D_63_XM	D_63_XL	D_64_O	D_64_U	D_64_R	D_64_I	target
0	1	0	0	0	0	0	0	1	0	0	0	0
1	1	0	0	0	0	0	0	0	1	0	0	0
2	1	0	0	0	0	0	0	1	0	0	0	1
3	1	0	0	0	0	0	0	1	0	0	0	0
4	0	0	1	0	0	0	0	1	0	0	0	1
...	...	...	...	...	...	...	...	...	...	...	...	...
6036	1	0	0	0	0	0	0	0	1	0	0	0
6037	1	0	0	0	0	0	0	0	1	0	0	1
6038	1	0	0	0	0	0	0	0	0	1	0	1
6039	1	0	0	0	0	0	0	0	1	0	0	0
6040	0	1	0	0	0	0	0	1	0	0	0	0

6041 rows × 12 columns

Figura 4: Manejo de variables categóricas

Luego, procedimos a analizar el resto de las variables para verificar si contenían valores atípicos, con el fin de determinar el método más apropiado para completar los valores faltantes.
Si se identificaban datos atípicos y, después de examinar la distribución de los datos, se determinaba que llenar los valores faltantes con la moda era la mejor opción para evitar una desviación significativa.



Cuadro 5: Análisis de los grupos

Después de la codificación y el llenado de los espacio na, el archivo está listo para usarse:

```
[35] print(f'Informacion faltate en el DF: {df.isna().sum().sum()}')

     Informacion faltate en el DF: 0
```

3.2. Modelos supervisados

Siguiendo las instrucciones dadas para el proyecto, hicimos iteraciones para 4 modelos supervisados:

Logistic Regression

Descripción:
Logistic Regression es un modelo de regresión utilizado para problemas de clasificación binaria. Estima la probabilidad de que una observación pertenezca a una clase específica.
Funcionamiento:
Utiliza la función logística para modelar la probabilidad de que una instancia pertenezca a una categoría particular. Ajusta los parámetros basándose en la función de pérdida y puede extenderse para tareas de clasificación multiclase.

Random Forest:

Descripción: Random Forest es un algoritmo de aprendizaje ensemble que construye múltiples árboles de decisión y los combina para obtener predicciones más robustas.
Funcionamiento: Crea múltiples árboles de decisión utilizando subconjuntos aleatorios de datos y características. Luego, combina las predicciones de cada árbol para producir una predicción final. Es conocido por su eficacia en la clasificación y regresión.

SVM (Support Vector Machine):

Descripción: SVM es un modelo de aprendizaje supervisado utilizado para clasificación y regresión.
Funcionamiento: Busca el hiperplano que mejor separa las clases maximizando el margen entre las clases. Puede utilizar diferentes kernels para trabajar con datos no lineales.

XGBoost:

Descripción: XGBoost es un algoritmo de boosting que se ha destacado en competiciones de aprendizaje automático por su eficiencia y precisión.
Funcionamiento: Combina varios árboles débiles en un único modelo más fuerte. Utiliza gradient boosting para optimizar la función de pérdida y es conocido por su rapidez y capacidad para lidiar con grandes conjuntos de datos.

3.3. Modelos no supervisados

PCA (Principal Component Analysis):

Descripción: PCA es una técnica de reducción de dimensionalidad utilizada para comprimir la información de un conjunto de variables correlacionadas en un conjunto más pequeño de variables no correlacionadas.
Funcionamiento: Encuentra una proyección de los datos originales en un espacio de menor dimensión que capture la mayor variabilidad posible en los datos.

Combinación PCA - Random Forest / XGBoost:

Descripción: La combinación de PCA con modelos de aprendizaje supervisado como Random Forest y XGBoost ayuda a reducir la dimensionalidad de los datos antes de aplicar los modelos, lo que puede mejorar la eficiencia y la precisión en ciertos casos.

3.4. Resultados, métricas y curvas de aprendizaje.

Logistic Regression

Métricas

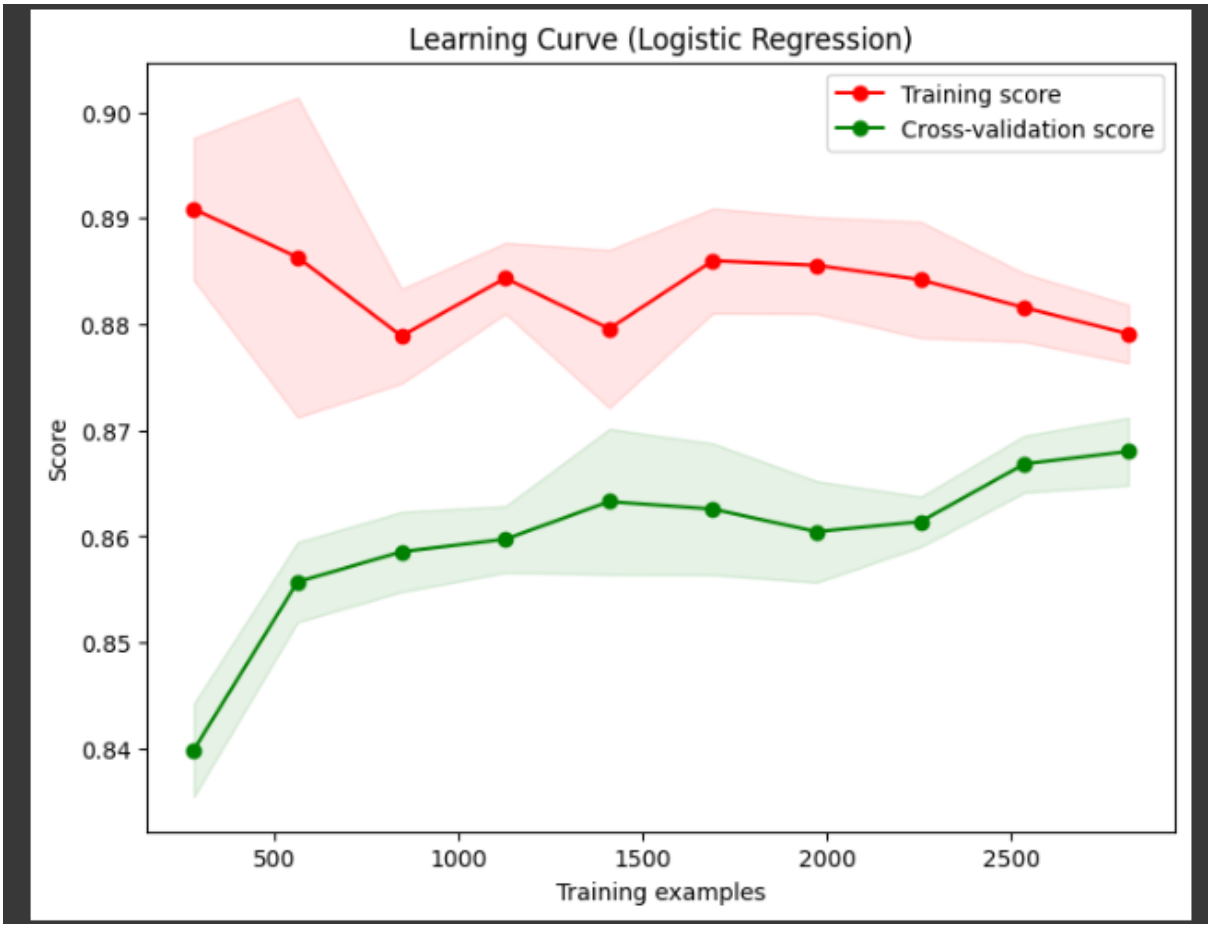
```
Logistic Regression Metrics:
              precision    recall  f1-score   support

           0       0.90      0.93      0.92      1385
           1       0.75      0.67      0.71       428

    accuracy                           0.87      1813
   macro avg       0.83      0.80      0.81      1813
weighted avg       0.87      0.87      0.87      1813

Accuracy: 0.8692774407060121
ROC AUC Score: 0.7998161206518438
```

Curva de aprendizaje



Resultados

-Según las métricas podemos ver que en general el modelo presenta buenos resultados, con un accuracy del 87% y un recall alto, lo que significa que está proyectando bien los datos. Por parte de la gráfica, podemos observar que el training score y el cross-validation score parecen converger a un mismo valor mientras se aumentan los datos, esto indica que el modelo generaliza bien y se beneficia de más datos.

Recomendaciones

-Considerar la adición de más datos si es posible. Esto podría ayudar a mejorar el rendimiento del modelo.

Random Forest

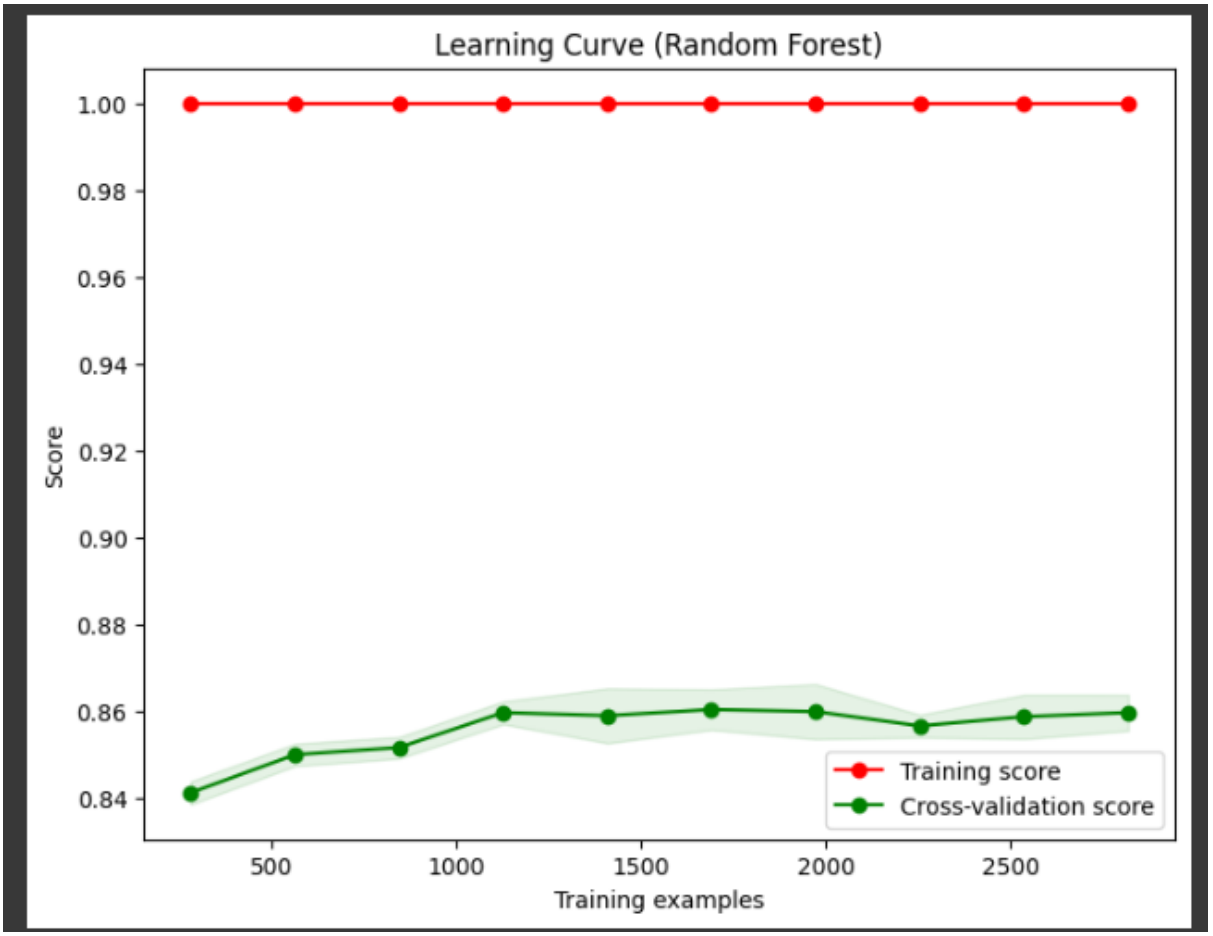
Métricas

```
Precisión del modelo: 0.8753447324875896
Reporte de clasificación:
              precision    recall  f1-score   support

           0       0.91      0.93      0.92      1385
           1       0.76      0.69      0.72       428

    accuracy                           0.88      1813
   macro avg       0.83      0.81      0.82      1813
weighted avg       0.87      0.88      0.87      1813
```

Curva de aprendizaje



Resultados

-Según las métricas podemos ver que en general el modelo presenta buenos resultados, con un accuracy del 88% y un recall bastante alto, lo que significa que está proyectando bien los datos. Por parte de la gráfica, podemos observar que el training score es alto y se mantiene alto a medida que se agregan más datos, lo que puede indicar que el modelo está capturando bien el patrón de los datos de entrenamiento, pero hay una brecha significativa entre el training score y el cross-validation score, indica que el modelo puede estar sobreajustando (alta puntuación de entrenamiento pero menor en validación cruzada) o subajustando (baja puntuación en ambos).

Recomendaciones

-Considerar el uso de técnicas de regularización (como L1, L2) para reducir el overfitting.
-Evaluar la calidad y relevancia de las características utilizadas en el modelo. Explorar técnicas de selección de características o ingeniería de nuevas características que sean más informativas para mejorar el rendimiento del modelo sin aumentar la complejidad.
-Simplificar el modelo podría ser beneficioso. Eliminar características irrelevantes.

XGBoost

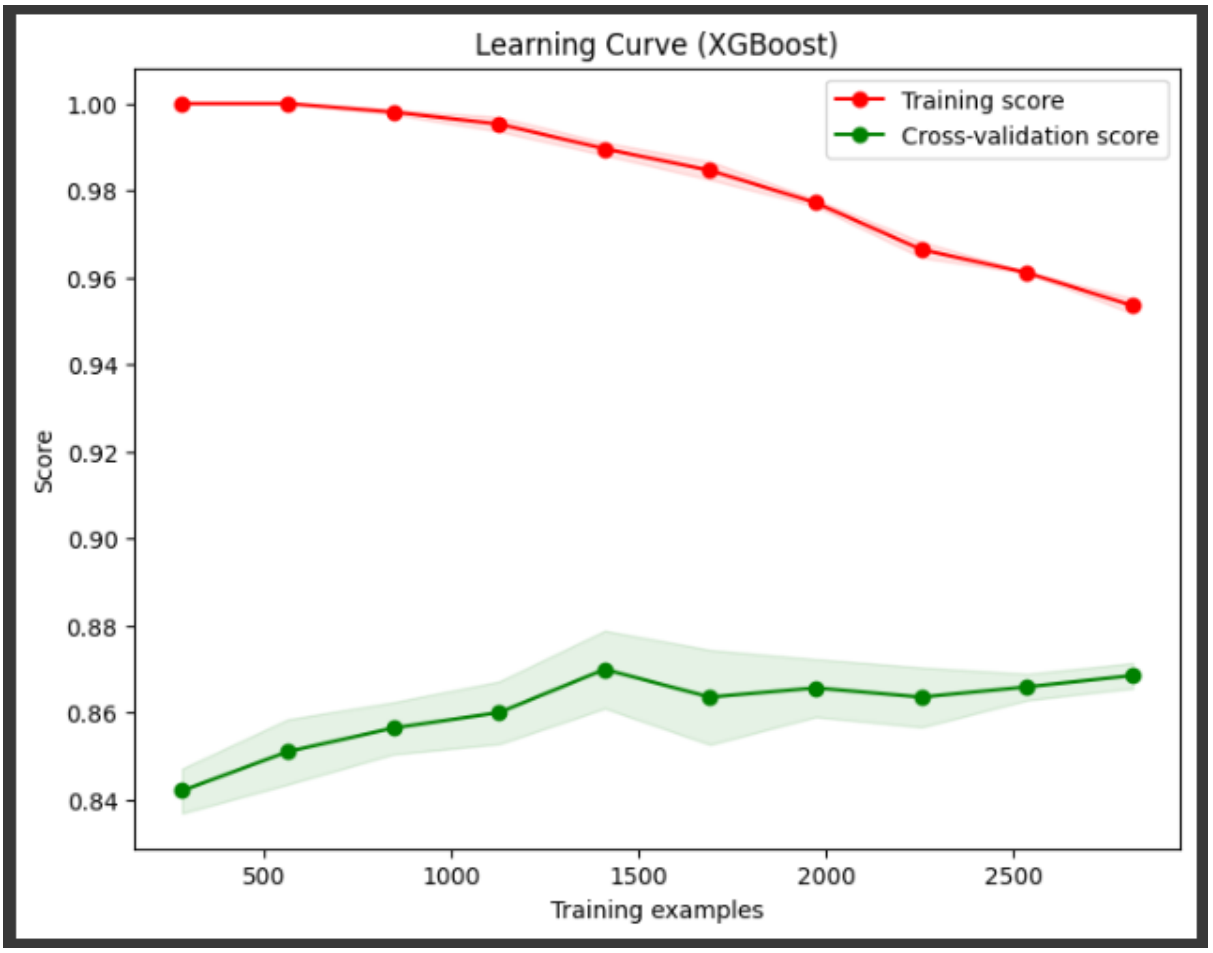
Métricas

```
Precisión del modelo: 0.8720353006067292
Reporte de clasificación:
              precision    recall  f1-score   support

           0       0.91      0.92      0.92      1385
           1       0.74      0.71      0.72       428

    accuracy                           0.87      1813
   macro avg       0.82      0.82      0.82      1813
weighted avg       0.87      0.87      0.87      1813
```

Curva de aprendizaje



Resultados

-Según las métricas podemos ver que en general el modelo presenta buenos resultados, con un accuracy del 87% y un recall alto, lo que significa que está proyectando bien los datos. Por parte de la gráfica, podemos observar que el training score y el cross-validation score parecen converger a un mismo valor mientras se aumentan los datos, esto indica que el modelo generaliza bien y se beneficia de más datos.

Recomendaciones
-Considerar la adición de más datos si es posible. Esto podría ayudar a mejorar el rendimiento del modelo.

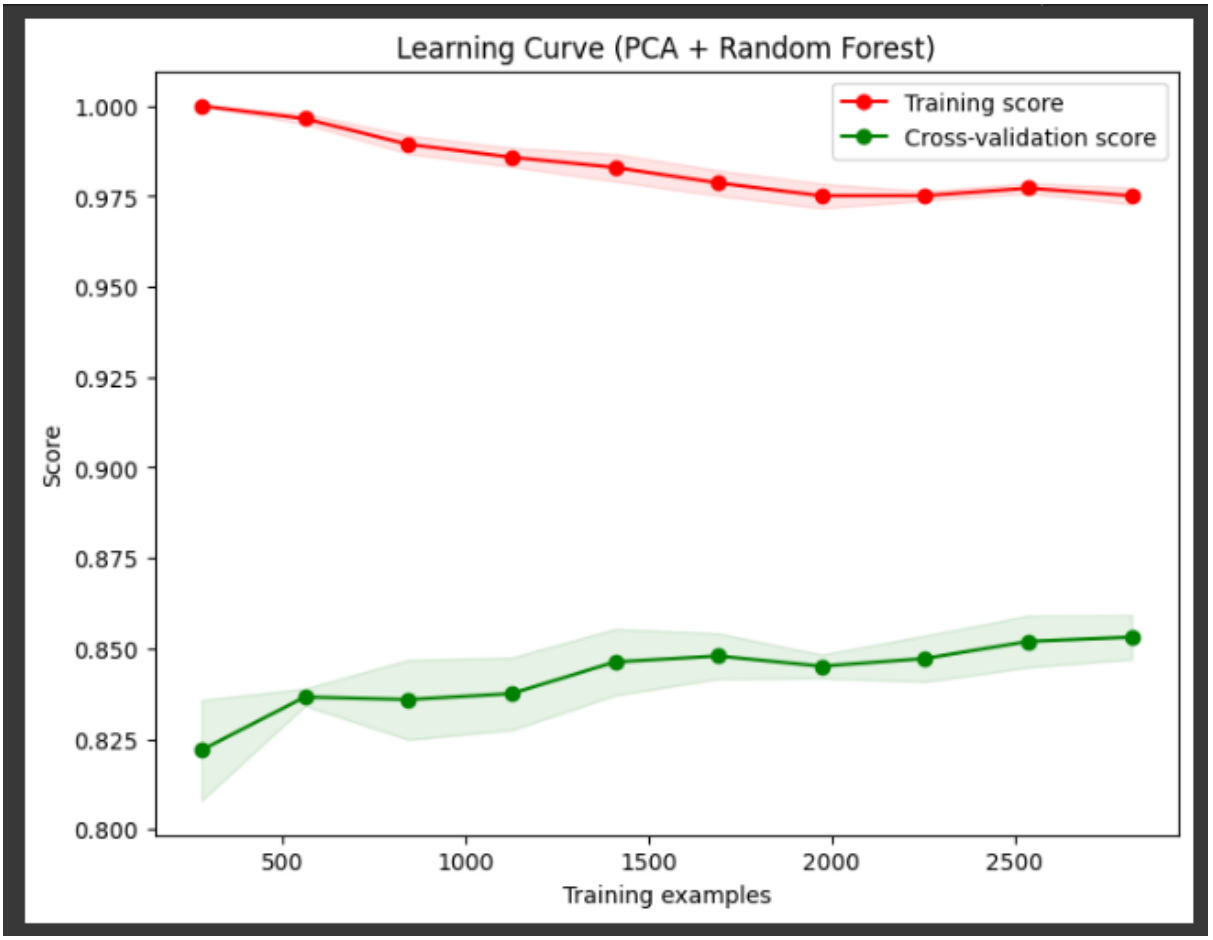
PCA-Random Forest

Métricas

Classification Report:

	precision	recall	f1-score	support
0	0.88	0.93	0.91	1385
1	0.73	0.60	0.66	428
accuracy			0.85	1813
macro avg	0.81	0.76	0.78	1813
weighted avg	0.85	0.85	0.85	1813

Curva de aprendizaje



Resultados
-Según las métricas podemos ver que en general el modelo presenta buenos resultados, con un accuracy del 85% y un recall bastante alto, lo que significa que está proyectando bien los datos. Por parte de la gráfica, podemos observar que el training score y el cross-validation score parecen converger a un mismo valor mientras se aumentan los datos, esto indica que el modelo generaliza bien y se beneficia de más datos.

Recomendaciones
-Considerar la adición de más datos si es posible. Esto podría ayudar a mejorar el rendimiento del modelo.

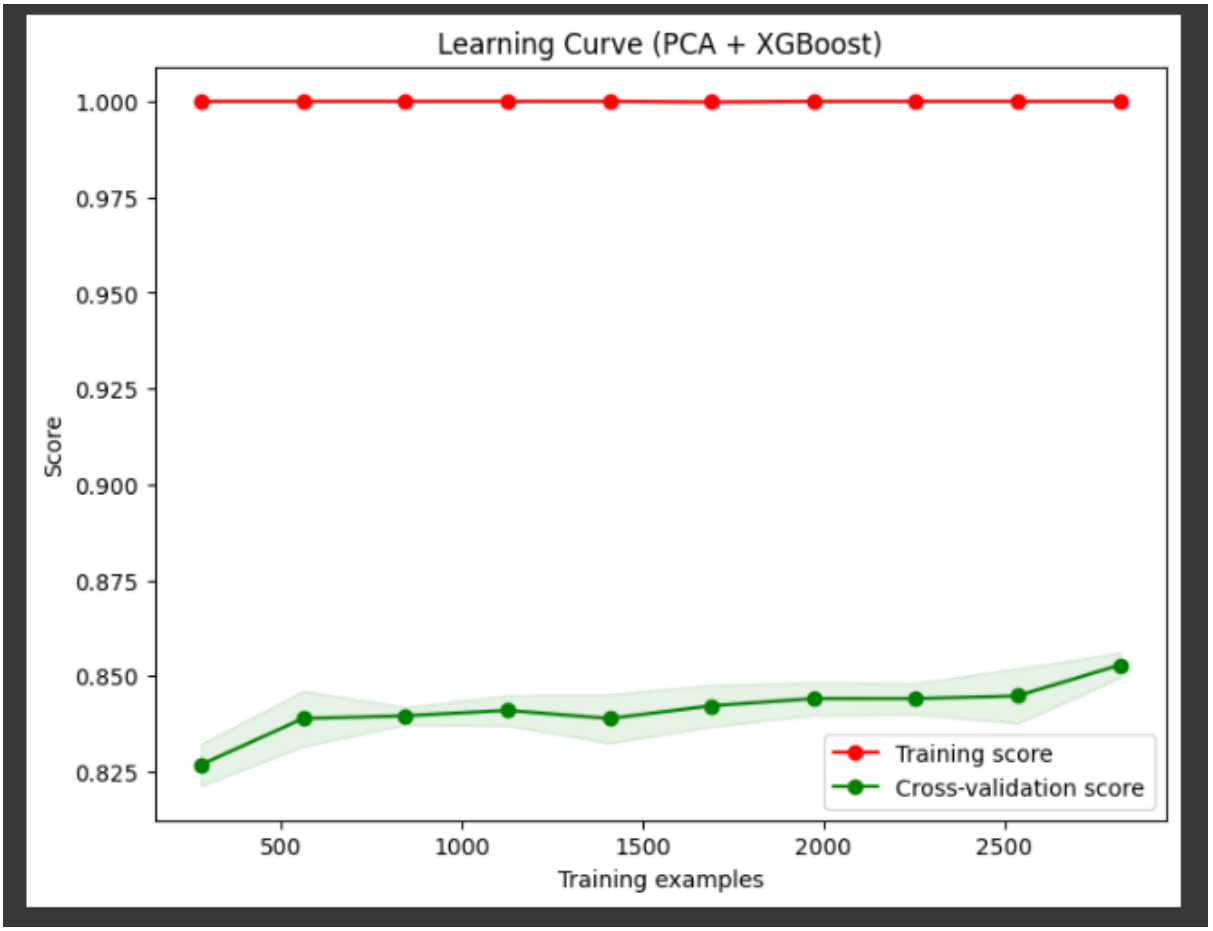
PCA-XGBoost

Métricas

Classification Report:

	precision	recall	f1-score	support
0	0.89	0.92	0.91	1385
1	0.72	0.64	0.68	428
accuracy			0.86	1813
macro avg	0.81	0.78	0.80	1813
weighted avg	0.85	0.86	0.86	1813

Curva de aprendizaje



Resultados
-Según las métricas podemos ver que en general el modelo presenta buenos resultados, con un accuracy del 86% y un recall bastante alto, lo que significa que está proyectando bien los datos. Por parte de la gráfica, podemos observar que el training score es alto y se mantiene alto a medida que se agregan más datos, lo que puede indicar que el modelo está

capturando bien el patrón de los datos de entrenamiento, pero hay una brecha significativa entre el training score y el cross-validation score, indica que el modelo puede estar sobreajustando (alta puntuación de entrenamiento pero menor en validación cruzada) o subajustando (baja puntuación en ambos).

Recomendaciones

-Considerar el uso de técnicas de regularización (como L1, L2) para reducir el overfitting.
-Evaluar la calidad y relevancia de las características utilizadas en el modelo. Explorar técnicas de selección de características o ingeniería de nuevas características que sean más informativas para mejorar el rendimiento del modelo sin aumentar la complejidad.
-Simplificar el modelo podría ser beneficioso. Eliminar características irrelevantes.

RETOS Y CONSIDERACIONES DE DESPLIEGUE

Umbral de desempeño mínimo

El umbral de desempeño mínimo para desplegar el modelo en producción se basará principalmente en el objetivo principal del proyecto, que es identificar a los clientes que no pagarán a tiempo (incumplimientos) con un alto recall. Garantizar un recall lo suficientemente alto para capturar la mayoría de los incumplimientos, reduciendo así el riesgo de pérdidas financieras para el banco American Express. El recall objetivo para el modelo en producción deberá ser del 85% cómo mínimo.

Despliegue en Producción

Para esto, se debe realizar una evaluación exhaustiva del modelo y su desempeño. Esto incluirá la validación del modelo en un entorno de producción simulado para asegurarse de que cumple con los requisitos específicos y objetivos del negocio. Además, se deben implementar prácticas sólidas de control de calidad para garantizar la estabilidad y consistencia del modelo en un entorno operativo. El despliegue también implica la integración del modelo en el flujo de trabajo existente, asegurando que sea fácilmente accesible y utilizable por los usuarios finales, y que se sigan protocolos de seguridad más adecuados para proteger los datos sensibles.

Monitoreo del desempeño en producción

Esto es bastante importante para garantizar que el modelo mantenga su precisión y fiabilidad a lo largo del tiempo. Implica el establecimiento de un sistema de monitoreo continuo que rastree métricas clave, como el recall, precisión, entre otros, en tiempo real o en intervalos regulares. Se deben definir alertas para detectar cualquier degradación en el rendimiento del modelo, lo que puede ser indicativo de cambios en los datos o problemas con el modelo mismo. También incluir la revisión periódica y la reevaluación del modelo, lo que puede llegar a implicar su reentrenamiento con datos actualizados para mantener su efectividad a lo largo del tiempo.

CONCLUSIONES

-Dado el problema inicial, pudimos lograr ver lo complejo que puede ser un sistema crediticio y todas las variables que lo afectan. Y con el desarrollo del proyecto se puede vislumbrar lo útil que puede llegar a ser un modelo de inteligencia artificial para predecir todo tipo de variables, en este caso, la probabilidad de que un cliente no pague el saldo de su tarjeta de crédito en el futuro.

-Pudimos observar cómo funcionan algunos de los modelos más usados para entrenar, cómo lo fueron Logistic Regression, Random Forest y XGBoost. Entender sus funcionalidades y las ventajas y desventajas que pueden traer cada uno de ellos a un problema determinado, en nuestro caso, un problema crediticio.

-Entender lo importante que es la toma de hiperparametros para cada modelo, ya que su rendimiento aumenta en gran medida teniendo en cuenta los parámetros adecuados.

-Lo importante que es hacer un buen diagnóstico de desempeño a los diferentes modelos, a través de sus métricas y gráficas, para determinar cuál modelo se ajusta mejor a nuestras circunstancias. Cómo algunos modelos, a pesar de presentar unos buenos resultados en las métricas de desempeño, después de ver su curva de aprendizaje, podemos notar que presentan overfitting, y las medidas a tomar en estos casos.

-Explorar los desafíos y condiciones para llevar los modelos a producción. Cómo puede ser difícil determinar el nivel mínimo de desempeño necesario para la implementación, el proceso de despliegue en producción y las estrategias de monitoreo y mantenimiento, si no se tiene claro cuál es el problema a resolver.

-Aprendimos a grandes rasgos cómo es todo el proceso para entrenar a una inteligencia artificial, los retos que estos conllevan y los aprendizajes que se pueden obtener en el camino.

Video explicatorio

BIBLIOGRAFÍA

-Addison Howard, AritraAmex, Di Xu, Hossein Vashani, inversion, Negin, Sohier Dane. (2022). American Express - Default Prediction. Kaggle. https://kaggle.com/competitions/amex-default-prediction

-Gomez Juan.(2023).Métricas De Evaluación De Modelos En El Aprendizaje Automático. DataSource.ai. https://www.datasource.ai/es/data-science-articles/metricas-de-evaluacion-de-modelos-en-el-aprendizaje-automático

-Amat Joaquin.(2023). Random Forest con Python. CienciaDeDatos.net. https://cienciadedatos.net/documentos/py08_random_forest_python

-Ramos Raul.(2020). Inteligencia Artificial para las Ciencias e Ingenierías. Universidad de Antioquia. https://rramosp.github.io/ai4eng.v1/intro.html

-Bosco Juan.(2020). Tutorial: XGBoost en Python. medium.com. https://medium.com/@jboscomendoza/tutorial-xgboost-en-python-53e48fc58f73

-Recurero de los Santos Paola.(2018). Python para todos: Tutorial de PCA en 5 sencillos pasos. Telefonia Tech. https://telefonicatech.com/blog/python-para-todos-tutorial-de-pca-en-5