UNIVERSIDAD AUTÓNOMA DE NUEVO LEÓN

FACULTAD DE CIENCIAS FÍSICO MATEMÁTICAS

MAESTRÍA EN CIENCAS DE DATOS

APRENDIZAJE AUTOMÁTICO

MAESTRO: JOSÉ ALBERTO BENAVIDES VÁZQUEZ

PROYECTO

ALUMNO: EDWIN MARTÍN ROMERO SILVA

MATRÍCULA: 1731276

Índice

1.-Base de datos

La base de datos que escogí se llama **aplication_record**, está relacionada con el crédito.

Contiene 20,508 registros de tarjetas de crédito y cada registro contiene información del cliente al momento de la solicitud de crédito. Luego de realizar limpieza en la base de datos y crear nuevas variables a partir de las ya existentes, la base de datos contiene 21 variables finales.

Categóricas (5):

- Cat_Tipo_Ingresos
- Cat_Ocupacion
- Cat_Educacion
- Cat_Edo_Civil
- Cat_Vivienda

Binarias (7)

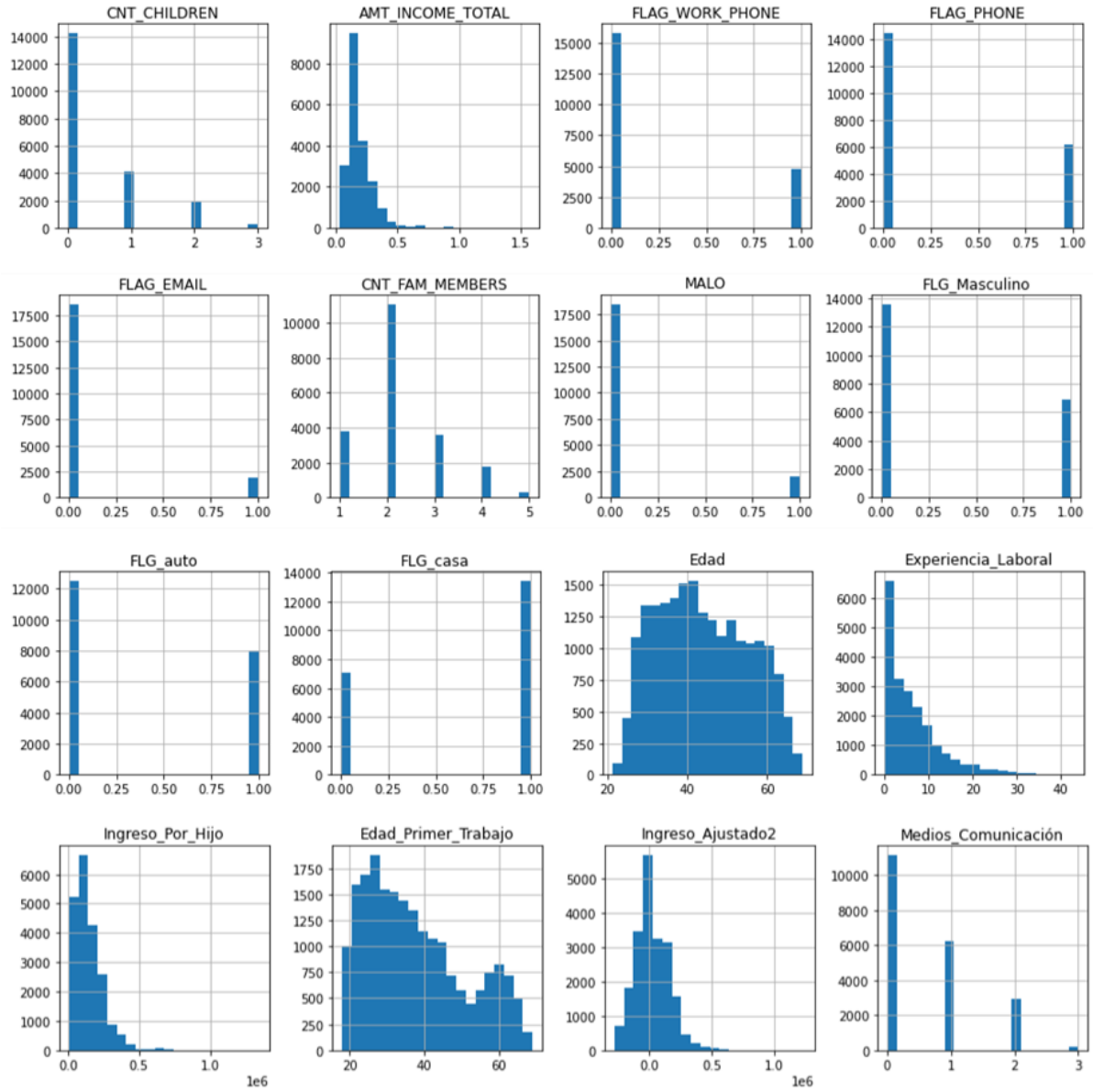
- FLAG_WORK_PHONE
- FLAG_PHONE
- FLAG_EMAIL
- FLG_Masculino
- FLG_auto
- FLG_casa
- MALO

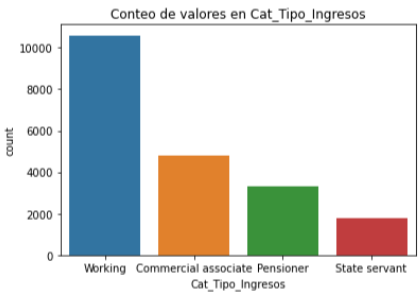
Continuas o discretas (9)

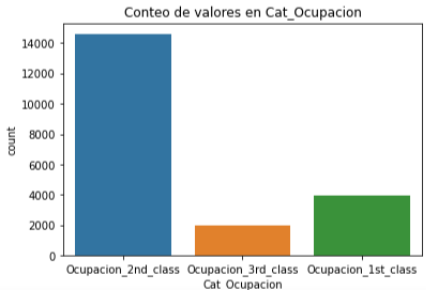
- CNT_CHILDREN
- AMT_INCOME_TOTAL
- CNT_FAM_MEMBERS
- Edad
- Experiencia_Laboral
- Ingreso_Por_Hijo
- Edad_Primer_Trabajo
- Ingreso_Ajustado2
- Medios_Comunicacion

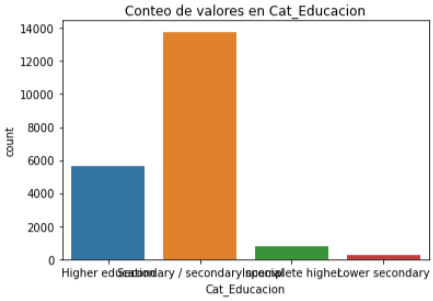
Obtuve esta base desde la plataforma de Kaggle.

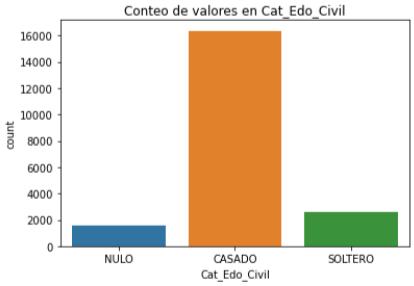
2.-Estadística Descriptiva

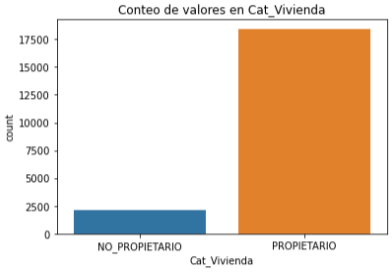












3.-Variable Objetivo

La variable objetivo de mi proyecto es la variable **MALO**, esto quiere decir que es la variable que buscaré predecir con mi modelo.

Esta variable es binaria (0/1) y sirve para indicar el desempeño de una tarjeta de crédito. De forma que podemos identificar con un 1 a los clientes malos y con un 0 a los clientes buenos.

Un cliente se considera 'malo' con lo siguiente: Atraso de 60 días o más, durante los primeros 12 meses desde que se coloca la tarjeta de crédito.

En mi experiencia, 12 meses de desempeño es una cantidad suficiente para determinar si un cliente titular de una tarjeta de crédito se puede clasificar como Bueno o Malo. Ya que es el tiempo promedio que tarda en 'madurar' una cosecha de tarjetas de crédito.

En esta base de datos cerca del 10% de los clientes están marcados como malos.

MALO	Registros	%Registros
0	18491	0.901648
1	2017	0.098352

4.-Modelo Elegido

Escogí el Xtreme Gradient Boosting para mi problema, usé la versión XGB Classifier, ya que mi problema es de clasificación. Recordemos que el objetivo es predecir la variable binaria MALO.

El XGB es un algoritmo de aprendizaje automático supervisado que se basa en la técnica de boosting.

El boosting es una técnica de ensamble en la que se combinan N modelos simples (generalmente árboles de decisión) para crear un modelo más fuerte. El objetivo principal del boosting es mejorar el rendimiento del modelo, disminuyendo los errores del modelo anterior en iteraciones sucesivas.

Es decir, el XGB funciona entrenando N modelos simples enfocándose en corregir los errores del modelo anterior y enfocandose en las instancias que el modelo previo clasificó incorrectamente.

Este algoritmo es ampliamente utilizado y es conocido por su eficiencia y capacidad para manejar una variedad de problemas de regresión y clasificación.

5.-Métricas de Desempeño

Las métricas de error o de desempeño nos permiten medir que tan bueno es nuestro modelo para cumplir su objetivo y de esta forma tomar decisiones.

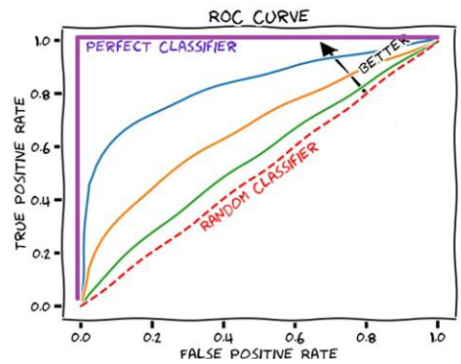
Yo escogí el **AUC (Área bajo la curva)** ya que se adapta perfectamente a mi problema de clasificación binaria, además de que es uno que utilizo en el trabajo y estoy más familiarizado con él.

El AUC funciona de la siguiente manera:

Primero, la curva ROC es una representación gráfica que se utiliza comúnmente en problemas de clasificación para evaluar el rendimiento de un modelo, particularmente en modelos binarios. Esta curva muestra la relación entre dos tasas:

- **Tasa de Verdaderos Positivos (TPR)**: Mide la mide la capacidad del modelo para identificar instancias positivas. Esta es la proporción de positivos reales que el modelo clasifica correctamente como positivos.
- **Tasa de Falsos Positivos (FPR)**: Mide la tasa de errores en los que el modelo clasifica incorrectamente instancias negativas como positivas. Esta es la proporción de negativos reales que el modelo clasifica incorrectamente como positivos.

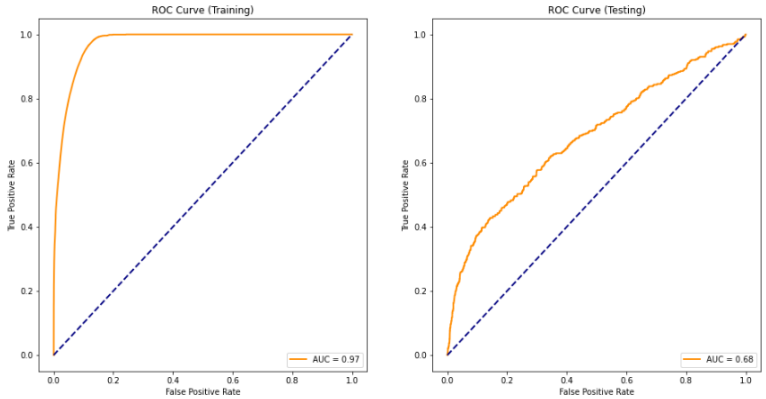
La curva ROC traza la TPR en el eje vertical y la FPR en el eje horizontal a medida que se ajusta el umbral de decisión del modelo, tal como se muestra en la siguiente gráfica:



El AUC es el área debajo de la curva ROC. Cuanto mayor sea el AUC, mejor será el rendimiento del modelo. El AUC varía de 0 a 1, donde:

- AUC menor a 0.5 indica que el modelo es peor que una decisión aleatoria.
- AUC igual a 0.5 indica que el modelo tiene un rendimiento similar al aleatorio.
- AUC mayor a 0.5 indica que el modelo tiene cierta capacidad de discriminación y es mejor que una decisión aleatoria.
- AUC igual a 1 indica un clasificador perfecto que clasifica todas las instancias positivas antes que las negativas.

Para ilustrar el uso de la métrica en mi proyecto, calculé la curva ROC en mi conjunto de datos para evaluar el rendimiento del XGB Classifier. Este análisis se realizó tanto en la muestra de entrenamiento (70%) como en la muestra de prueba (30%), utilizando los parámetros por defecto del modelo.



Resulta evidente que el AUC en la muestra de entrenamiento supera al de la muestra de prueba. Esta discrepancia sugiere que, con los parámetros por defecto, el modelo exhibe un mejor rendimiento en la muestra de entrenamiento.

Es relevante mencionar que opté por utilizar los parámetros por defecto en este ejemplo. Cualquier ajuste adicional o consideración sobre la elección de estos parámetros se detalla en la sección de **Diseño de Experimentos.**

6.-Entrenamiento y Validación

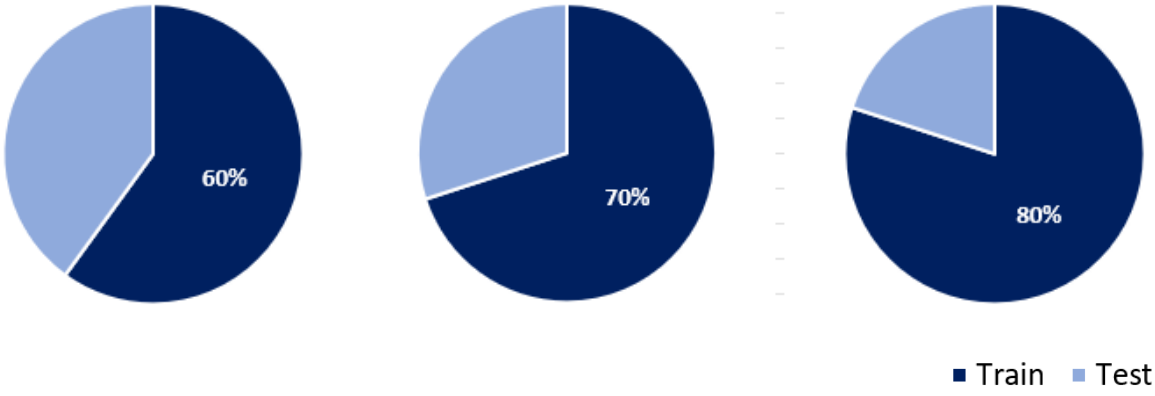
Uno de los problemas a los que nos enfrentamos al entrenar un modelo es el sobreajuste.

Se dice que un modelo presenta sobreajuste cuando tiene un gran desempeño al predecir en la base de datos de entrenamiento, pero no lo hace de la misma manera en datos nuevos.

Para reducir el sobre ajuste realizaremos experimentos con nuestro modelo dividiendo nuestros datos en 2 muestras, la de entrenamiento y la de validación. Buscando elegir la combinación de hiperparámetros que nos permitan alcanzar un buen desempeño tanto en la muestra de entrenamiento como en la de validación.

A través del diseño de experimentos, utilizaremos 3 combinaciones de porcentajes para segmentar nuestra base de datos, con la hipótesis de que en 1 de esas 3 combinaciones se reduzca de menor manera el sobre ajuste:

- 60% y 40%
- 70% y 30%
- 80% y 20%



7.-Diseño de Experimentos

7.1.- Parte 1

El diseño de experimentos de mi proyecto se enfocará en encontrar la mejor combinación de hiperparametros y porcentaje de muestra de entrenamiento que maximiza el desempeño del modelo XGB en la muestra Test, ya que quiero encontrar el mejor modelo que sea capaz de predecir en datos que nunca ha visto.

Para esto, cree una matriz con distintas combinaciones de hiperparametros:

- **Estimadores = 25, 50, 75, 100**
  Este hiperparametro hace referencia a la cantidad de árboles que tendrá nuestro modelo, la teoría dice que una gran cantidad de árboles aumenta el desempeño, pero aumenta el costo computacional.

- **Subsample = 0.5, 0.7, 0.85, 1**
  Este hiperparametro indica el porcentaje de población que tomará cada árbol para entrenarse. La teoría dice que el no utilizar el 100% de la muestra para cada árbol, le da diversidad a cada uno de ellos, lo que podría reducir el sobreajuste.

- **max_depth = 3, 7, 12**
  Este hiperparametro hace referencia a la profundidad de cada árbol.

- **learning rate = 0.05, 0.1, 0.15**
  Este hiperparámetro controla la tasa de aprendizaje del modelo, es decir, la magnitud con la que los ajustes se realizan durante cada iteración del entrenamiento.

- **muestra train = 60%, 70%, 80%**
  Estos representan el porcentaje de la muestra de entrenamiento que se utilizará para entrenar el modelo XGB. Un mayor porcentaje puede llevar a un modelo más robusto, pero es importante equilibrar esto con la capacidad de cómputo disponible y evitar el sobreajuste en conjuntos de datos pequeños.

El resto de hiperparametros los dejé en su valor por Default.

	OP_ESTIMATORS	OP_SUBSAMPLE	OP_MAX_DEPTH	OP_LEARNING_RATE	OP_COLSAMPLE	OP_LOSS	OP_TRAIN_SAMPLE
0	25	0.5	3	0.05	1	log_loss	0.6
1	25	0.5	3	0.05	1	log_loss	0.7
2	25	0.5	3	0.05	1	log_loss	0.8
3	25	0.5	3	0.10	1	log_loss	0.6
4	25	0.5	3	0.10	1	log_loss	0.7
...	...	...	...	...	...	...	...
427	100	1.0	12	0.10	1	log_loss	0.7
428	100	1.0	12	0.10	1	log_loss	0.8
429	100	1.0	12	0.15	1	log_loss	0.6
430	100	1.0	12	0.15	1	log_loss	0.7
431	100	1.0	12	0.15	1	log_loss	0.8

432 rows × 7 columns

Probé 432 combinaciones distintas de hiperparámetros y obtuve su desempeño AUC tanto para la muestra Train como para la muestra Test e hice un análisis para determinar cuáles hiperparámetros son los que tienen mayor influencia en el desempeño de mi modelo:

OP_ESTIMATORS	AUC >= 0.65	AUC < 0.65
25	31	77
50	40	68
75	45	63
100	45	63
Total	161	271

OP_SUBSAMPLE	AUC >= 0.65	AUC < 0.65
0.5	36	72
0.7	44	64
0.85	47	61
1	34	74
Total	161	271

OP_MAX_DEPTH	AUC >= 0.65	AUC < 0.65
3	0	144
7	33	111
12	128	16
Total	161	271

OP_LEARNING_RATE	AUC >= 0.65	AUC < 0.65
0.05	47	97
0.1	54	90
0.15	60	84
Total	161	271

OP_TRAIN_SAMPLE	AUC >= 0.65	AUC < 0.65
0.6	33	111
0.7	57	87
0.8	71	73
Total	161	271

Mi conclusión de este experimento es que en mi base de datos, un mayor número de arboles nos da un mejor desempeño en nuestro modelo XGB (muestra test), además de una mayor profundidad, un mayor porcentaje en la muestra de entrenamiento y un mayor porcentaje de learning rate.

También parece tener un mejor desempeño en un subsample entre 0.7 y 0.85.

No elegiré el modelo de mejor desempeño en este primer experimento, si no que haré un segundo experimento, esta vez, acercándome un poco a los hiperparametros que mostraron un mejor desempeño, es decir:

- Estimadores > 75
- Max Depth > 12
- Porcentaje de entrenamiento = 80% (No aumentaré ya que dejar menos del 20% para la muestra de validación podría significar un riesgo)
- Learning rate > 0.15

7.2.- Parte 2

Repetí el experimento de la parte 1, esta vez con los siguientes hiperparametros

- **Estimadores = 75, 100, 125, 150**
- **Subsample = 0.7, 0.85**
- **max_depth = 12, 15, 20**
- **learning rate = 0.15, 0.2**
- **muestra train = 80%**

También repetí el análisis anterior para ver que hiperparametros son los que logran un mejor desempeño:

- Estimadores = 75
- Max Depth = 20
- Muestra de entrenamiento = 80%
- Subsample = 0.85
- Learning rate = 0.15

OP_ESTIMATORS	AUC >= 0.7	AUC < 0.7
75	6	6
100	4	8
125	4	8
150	4	8
Total	18	30

OP_SUBSAMPLE	AUC >= 0.7	AUC < 0.7
0.5	0	0
0.7	5	19
0.85	13	11
1	0	0
Total	18	30

OP_MAX_DEPTH	AUC >= 0.7	AUC < 0.7
12	5	11
15	5	11
20	8	8
Total	18	30

OP_LEARNING_RATE	AUC >= 0.7	AUC < 0.7
0.15	11	13
0.2	7	17
Total	18	30

De forma que no escogí el modelo con mejor desempeño en Test, escogí el modelo con los mejores hiperparámetros acorde mi análisis. El modelo con el séptimo mejor desempeño.

OP_ESTIMATORS	OP_SUBSAMPLE	OP_MAX_DEPTH	LEARNING_R	OP_TRAIN_SAMPLE	AUC_TRAIN	AUC_TEST
150	0.85	20	0.15	0.8	0.9756	0.7078
100	0.7	15	0.15	0.8	0.9743	0.7077
75	0.85	15	0.2	0.8	0.9749	0.7065
75	0.7	20	0.15	0.8	0.9743	0.7045
100	0.7	12	0.15	0.8	0.9729	0.7043
125	0.85	15	0.15	0.8	0.9752	0.7039
75	**0.85**	**20**	**0.15**	**0.8**	**0.9749**	**0.7034**
150	0.85	15	0.15	0.8	0.9759	0.7033
75	0.7	15	0.2	0.8	0.9742	0.7018
150	0.85	12	0.15	0.8	0.9754	0.7017
100	0.85	20	0.15	0.8	0.9753	0.7014
125	0.85	12	0.2	0.8	0.9750	0.7014
150	0.7	12	0.2	0.8	0.9747	0.7012
125	0.85	20	0.15	0.8	0.9755	0.7012
100	0.85	20	0.2	0.8	0.9760	0.7011
75	0.85	20	0.2	0.8	0.9755	0.7010
75	0.85	12	0.15	0.8	0.9719	0.7007

7.3.- Parte 3

Por último, tomé la combinación de hiperparámetros que escogí y entrené el modelo XGB 50 veces y saqué el promedio del desempeño en test de estas 50 iteraciones, para comprobar que en promedio, el desempeño se acercara al desempeño esperado de 0.7034

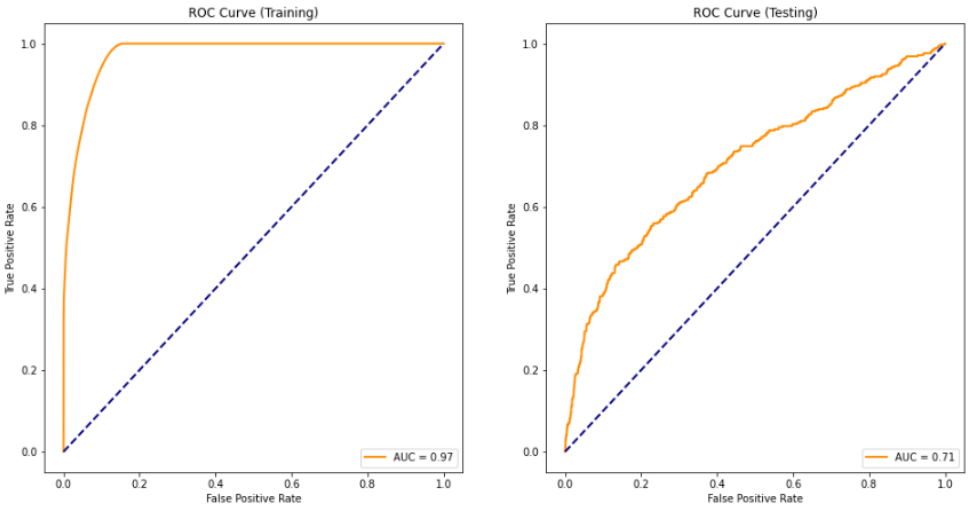
```
for j in range(50):
    df_train = df.sample(frac = 0.8)
    df_test = df.drop(df_train.index)

    y_train = df_train['MALO']
    x_train = df_train.drop(columns = ['MALO'])

    y_test = df_test['MALO']
    x_test = df_test.drop(columns = ['MALO'])

    # Crear un clasificador XGBoost
    XGBC = xgb.XGBClassifier(objective           = 'binary:logistic',
                             max_depth           = 20,
                             learning_rate       = 0.15,
                             n_estimators        = 75,
                             subsample           = 0.85,
                             colsample_bytree    = 1)
    XGBC.fit(x_train, y_train)
```

Obtuve un desempeño promedio de 0.7022 lo cuál se acerca a. desempeño esperado de 0.7034, además puedo concluir que es un desempeño aceptable para modelos de este tipo.

```
sum(lista_test)/len(lista_test)
```

```
0.7022997417727744
```

Área bajo la curva del modelo final:



8.-Conclusión

Luego de lo aprendido en este curso y en el desarrollo de mi proyecto puedo concluir lo siguiente:

- **Limpieza:**
  La limpieza de datos antes de entrenar un modelo de aprendizaje automático es esencial y puede tener un impacto significativo en el desempeño y la interpretabilidad del modelo.
  Los datos limpios y con el formato correcto son fundamentales para obtener resultados precisos y confiables. La presencia de datos incorrectos o inconsistentes puede llevar a un desempeño deficiente del modelo y a interpretaciones erróneas.
- **Desempeño:**
  Es esencial medir el desempeño de un modelo para garantizar predicciones precisas. Incluso si el modelo muestra un buen rendimiento inicial, debemos realizar un monitoreo continuo para verificar que el desempeño se mantenga estable, especialmente a medida que se acumulan más datos.
  Dada la variedad de métricas disponibles, es fundamental elegir aquellas que se ajusten mejor a las características específicas de tu problema.
  Por ejemplo, yo escogí el AUC, que es perfecto para un problema de clasificación binaria y lo utilicé para escoger la mejor combinación de hiperparámetros en la fase de experimentación.
- **El mejor modelo:**
  No existe el 'mejor algoritmo de machine learning'. El mejor algoritmo, es el que se adapta a nuestro problema y a nuestra base de datos.
  Tampoco existe una combinación mágica de hiperparametros que permite resolver todos los problemas, la mejor combinación es la que se adapta a nuestra base de datos.
  El mejor modelo y la mejor combinación de hiperparametros la debemos encontrar experimentando con nuestra base.

En este proyecto luego de realizar la debida limpieza a mi base de datos, elegir la métrica de desempeño adecuada y experimentar, logré obtener un modelo Xtreme Gradient Boosting capaz de clasificar entre clientes buenos y malos con un desempeño de 0.7034, lo cual es aceptable.