Proyecto Integrador

Avance 7

Resumen Ejecutivo

Dra. Grettel Barcelo Alonso

EQUIPO 21

Bernardo Mijangos Flores	**A01793654**
Dalia Isabel López Tapia	**A01245026**
David Valles Canedo	**A01191310**

16 de junio de 2024

ÍNDICE

Síntesis del problema

El proyecto tiene como objetivo principal desarrollar modelos de aprendizaje automático para predecir el estado del crédito de los clientes grupales de Alsol Contigo. Usamos datos históricos de registros demográficos y financieros de los clientes, como edad, estado civil, número de créditos y métricas crediticias. Se busca construir modelos efectivos que distingan entre créditos buenos y malos.

La empresa Alsol Contigo, fundada en 2009, tiene como misión ofrecer servicios financieros a microempresarios rurales en el sur de México. En 2015, se fusionó con Vendo Fácil, una empresa especializada en financiamiento automotriz, lo que permitió a Alsol Contigo absorber su cartera y consolidarse con un enfoque en microcréditos grupales y créditos individuales en diversas modalidades. Su misión es “Brindar soluciones financieras con compromiso social a través de un servicio ágil, seguro, confiable y adaptado a las necesidades de los clientes”.

La operación de Alsol Contigo se centra significativamente en el Estado de Chiapas, donde opera 23 agencia de crédito grupal, atendiendo a 12,385 clientes que representan el 76.6% de su cartera total. El proyecto actual de la empresa tiene como objetivos aumentar la retención de clientes del 79.77% al 85%, incrementar la cartera de 80 millones a 103 millones en 2024, reducir las pérdidas por castigo de 5.2 millones a 4.5 millones, y disminuir la rotación de personal del 34% especialmente en el área comercial, entre otras metas.

El proceso de crédito incluye la prospección de nuevos créditos o renovación de existentes, pasando por el análisis, aprobación, desembolso, gestión de cartera y liquidación. Intervienen asesores de créditos, coordinadores, cajeros, administradores, gerentes, coordinadores regionales y la dirección general grupal. Utilizan una aplicación móvil para la solicitud de créditos que envía información a un CRM, donde se consulta el historial crediticio y capacidad de pago antes de aprobar o rechazar la solicitud. Una vez aprobado, se programa el desembolso y se firma el contrato con plazos de 5 a 25 pagos catorcenales o 10 a 24 pagos semanales.

La gestión del crédito se realiza mediante un Core Financiero que calcula intereses, saldos diarios, pagos, liquidaciones y genera reportes. Sin embargo, el cobro de cuotas vencidas aún se realiza manualmente por asesor en campo, lo que puede demorar varios días, Mejorar la gestión de cobranza es crucial para alcanzar objetivos como la retención de clientes, reducción de morosidad y castigos, disminución de rotación de personal y reducción de costos.

Producto	GRUPAL
Etiquetas de fila	**Suma de TOTAL**
agosto	410,311.30
julio	465,087.33
junio	558,222.29
marzo	286,571.62
enero	514,529.40
septiembre	427,203.50
febrero	655,453.14
octubre	313,762.82
mayo	352,398.80
noviembre	391,736.48
diciembre	453,181.47
abril	415,045.47
Total general	**5,243,503.62**

Montos de castigo mensual del producto grupal

Aunque se disponen de herramientas que agilizan el proceso de crédito, aún se puede optimizar ciertos procesos, especialmente la gestión de cobranza. Actualmente, el cobro de cuotas vecindad se realiza mediante visitas programadas por los asesores de campo. Esta programación, que intenta alinearse con las fechas de vencimiento, a menudo enfrenta dificultades, lo que provoca retrasos en la gestión de cobranza. Estos retrasos incrementan la morosidad y afectan negativamente los indicadores de desempeño de los asesores, impactando

directamente en sus ingresos. Dado que se trata de créditos a corto plazo, es crucial implementar una gestión de cobranza más eficiente.

Para mejorar la retención de clientes, reducir la morosidad y los castigos, disminuir la rotación del personal y reducir los costos, se propone la implementación de un chatbot con Inteligencia Artificial utilizando el modelo open source “Llama2” de Meta. Este modelo, pre entrenado con datos públicos en línea, se ajustará mediante supervisión y se refinó interactivamente usando Reinforcement Learning from Human Feedback (RLHF), garantizando seguridad y efectividad en la gestión de los cobros y atención al cliente.

Se considera el desarrollo de un modelo supervisado para clasificar las actividades de los usuarios en categorías como historial, pagos y deudas, dudas técnicas y solicitudes de crédito, permitiendo así al chatbot elegir la mejor acción. Utilizaremos técnicas avanzadas de procesamiento de lenguaje natural, específicamente Llama2, para generar preguntas y respuestas manteniendo el contexto de la conversación. El modelo se ajusta específicamente al chatbot mediante fine-tuning para mejorar la interacción con los usuarios.

Hallazgos más importantes del análisis exploratorio de datos

La empresa cuenta con información histórica desde enero 2012 a la fecha, con 59,897 créditos con 4,232,529 pagos registrados, la base de datos cuenta con las siguientes variables:

- IdCliente: Identificador único del cliente.
- FechaAltaCliente: Fecha de registro como cliente.
- CodigoGrupo: Identificador único del grupo al que pertenece el cliente.
- EdadCliente: Edad del cliente al momento del desembolso.

- Genero: Femenino o Masculino
- EstadoCivil: Código del estado civil (1=SOLTERO(A), 2=CASADO(A), 3=VIUDO(A), 4=DIVORCIADO(A), 5=UNIÓN LIBRE, 6=SEPARADO, 7=SE IGNORA)
- Escolaridad: Código de la escolaridad (1=NO ESPECIFICADO, 1A=PRIMARIA COMPLETA, 1B=SECUNDARIA INCOMPLETA, 2A=SECUNDARIA COMPLETA, 2B=SECUNDARIA, 3=MÁS ESTUDIOS, 4=PREPARATORIA, 5=LICENCIATURA, 6=POSGRADO, 7A=NO SABE LEER NI ESCRIBIR, 7B=NO TIENE NIVEL EDUCATIVO, 8A=PRIMARIA INCOMPLETA, 8B=LEE Y ESCRIBE, 8C=SABE FIRMAR)
- CapacidadPago: Monto calculado mensual de ingresos menos egresos.
- Localidad: Código de la localidad.
- CicloCredito: Total de créditos con la empresa.
- CodigoSucursal: Código de la sucursal
- CodigoAgencia: Código de la agencia
- CodigoAsesor: Código del asesor de credito
- FechaDesembolso: Fecha del otorgamiento del crédito
- NumeroCredito: Número del crédito.
- FechaVencimiento: Fecha del vencimiento del crédito
- FechaCancelacion: Fecha de liquidación del crédito.
- Plazo: Número de pagos
- MontoCredito: Monto otorgado del crédito
- NumeroCuota: Número de cuota por crédito
- FechaVencimientoCuota: Fecha de pago de la cuota.
- FechaCancelaciónCuota: Fecha de liquidación de la cuota.
- MontoCuota: Importe total de la cuota.
- MontoCapita: Monto del capital.
- SaldoCapital: Saldo del capital
- EstatusCuota: Cancelado o Activo
- DiasAtraso: Días de atraso, diferencia en días de la variable FechaCancelacionCuota y FechaVencimientoCuota.

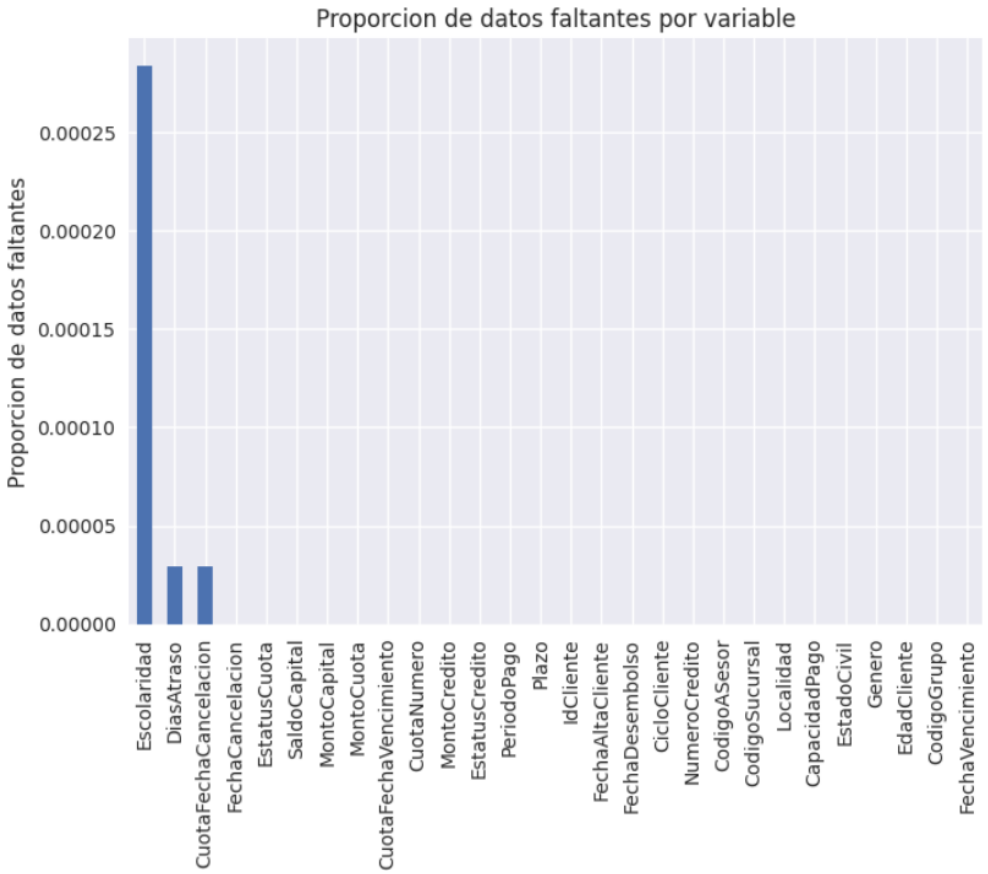
Análisis exploratorio

Realizamos la lectura de la información y lo asignamos a un dataframe

```
# Lectura del archivo
file_path = 'D:\Descargas\ProyectoIntegrador\DataSet\GrupalTodoCorregido.csv'

label = ["IdCliente","FechaAltaCliente","CodigoGrupo","EdadCliente","Genero","EstadoCivil","Escolaridad","CapacidadPago","Localidad","CodigoSucursal"

# Obtenemos el dataframe
grupal_df = pd.read_csv(file_path, names=label)

grupal_df.shape
```

```
(4382214, 28)
```

Buscamos y encontramos valores nulos de la base de datos, lo que se procede a eliminarlos ya que no son representativos del total de la muestra.

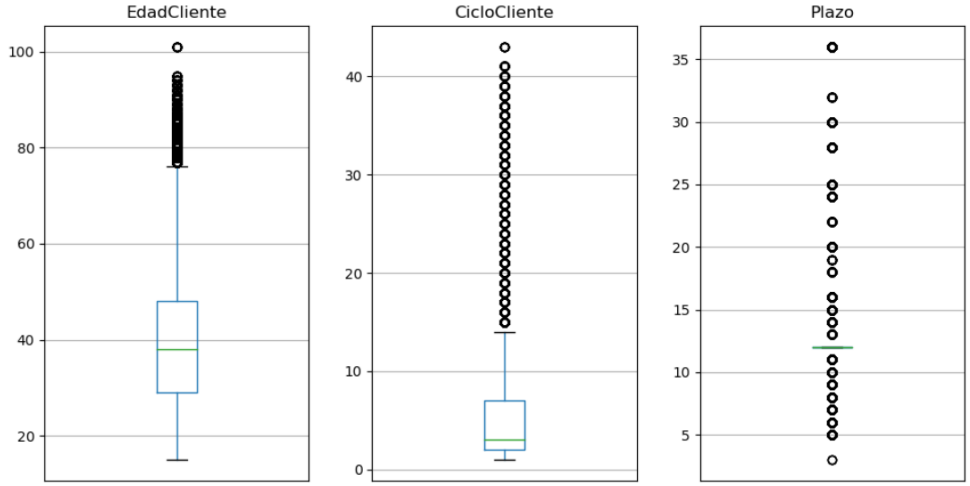


```
# eliminamos los valores nulos
gpdf = grupal_df.dropna()
```

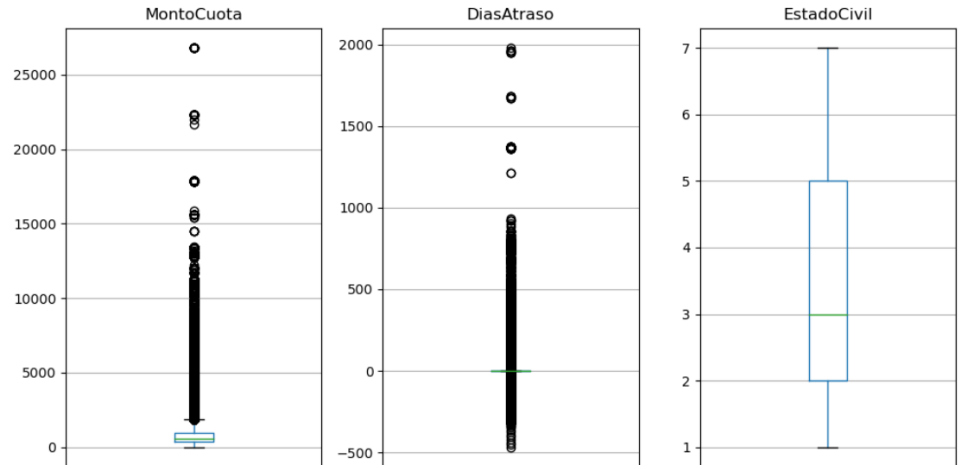
```
gpdf.isnull().values.any()
```

```
False
```

Se detecta que en la variable "EdadCliente" existen valores atípicos con más de 80 años, esto se debe que al momento de la consulta de la información se obtuvo el dato comparando la fecha de nacimiento con la fecha actual, cuando se debió tomar la fecha del desembolso que es el dato más real, para continuar con la exploración de los datos se procede a eliminar dichos registros.

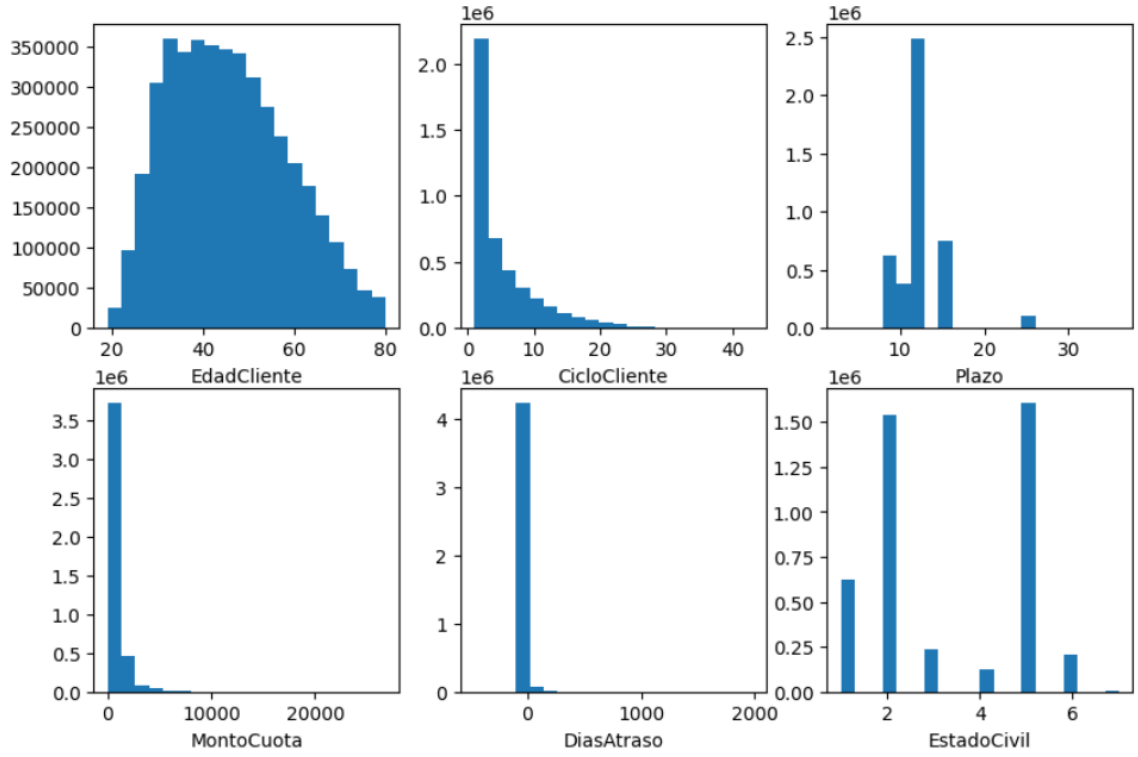


Así también se detecta que la variable DiasAtraso existen valores negativos, es consecuencia de la liquidación anticipada de la cuota.



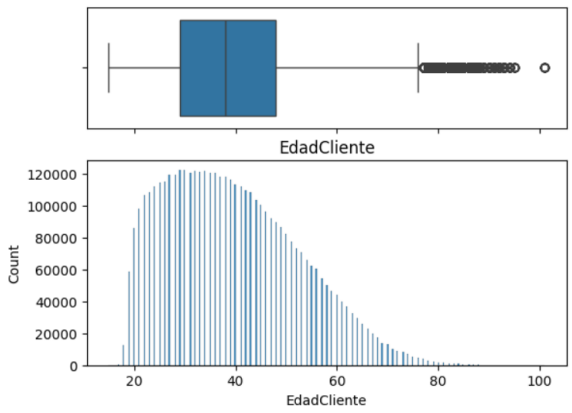
Realizamos la corrección de la consulta para tomar la edad correcta del cliente al momento del desembolso y para la variable DiasAtraso los valores negativos las reemplazamos con valor de 0.

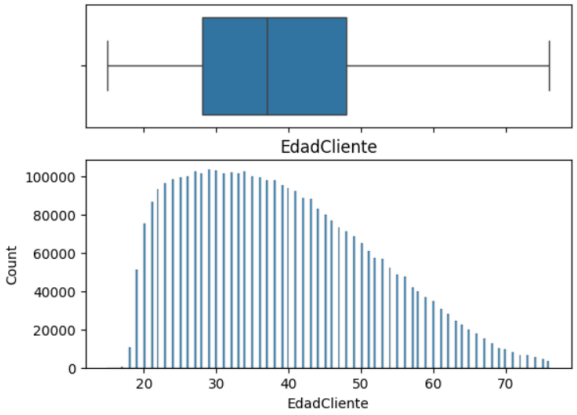
En los histogramas se observa que los valores sobre el eje horizontal de estas seis variables numéricas varían desde las unidades, hasta los miles; los histogramas de ClicloCliente, MontoCuota y DiasAtraso tienen un sesgo positivo; la variable EdadCliente tiene una distribución normal a excepción de la variable EstadoCivil que por ser una variable de tipo categórica proporciona una distribución muy sesgada.



Identificamos y tratamos los valores atípicos de la variable EdadCliente se determinan los límites inferior y superior de la variable utilizando la función find_limits, luego filtramos las observaciones para conservar sólo aquellas dentro de estos límites, posteriormente, se configura un outlierTrimmer para recortar los valores atípicos en variables usando el método del rango intercuartílico (IQR) y se ajusta a los datos. Visualizamos la distribución de EdadCliente antes y despues de

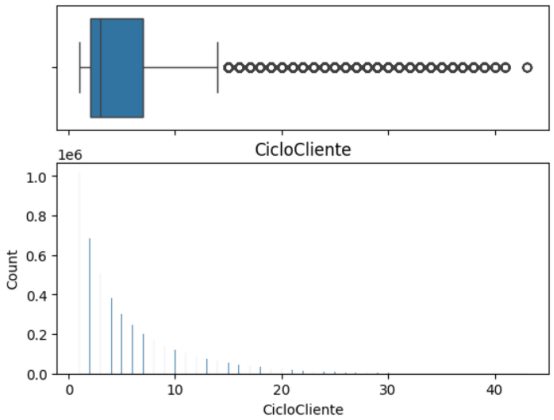
aplicar el recorte de valores atípicos, utilizando gráficos de caja e histogramas para evaluar los cambios de la distribución tras el tratamiento de los valores extremos.

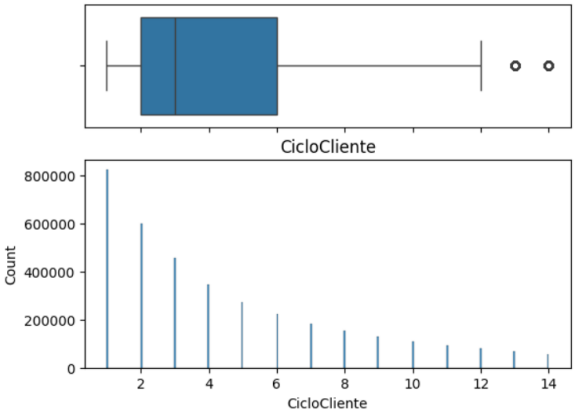




Antes de recortar los valores atípicos *Después del recorte de valores atípicos*

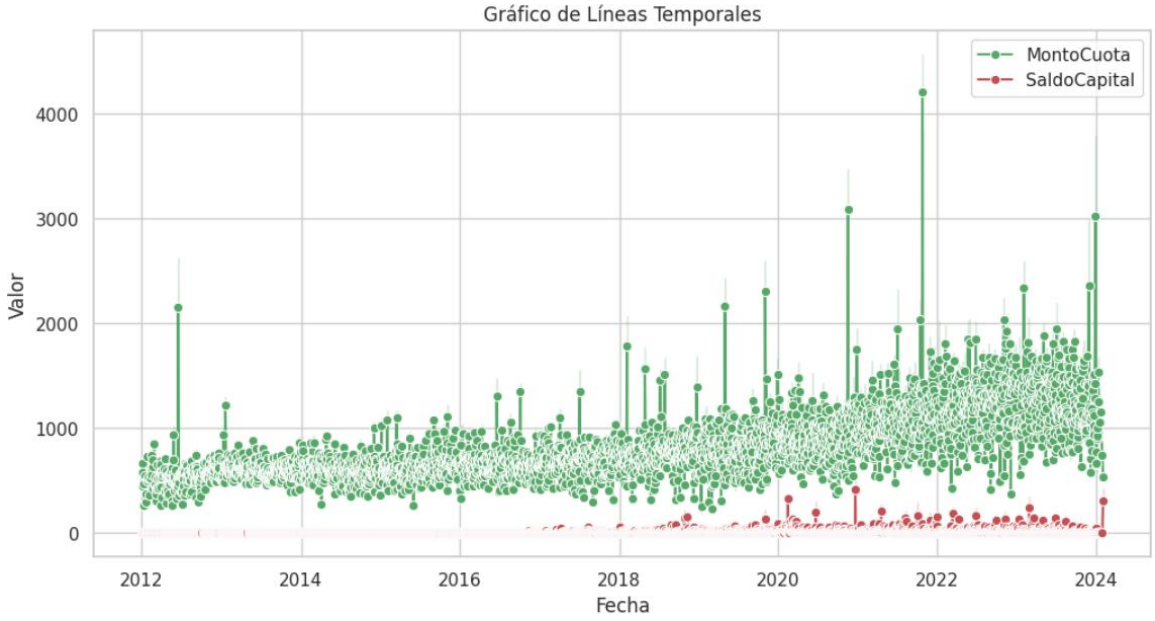
Realizamos los mismos recortes de valores atípicos con la variable CicloCliente obteniendo los siguientes gráficos.



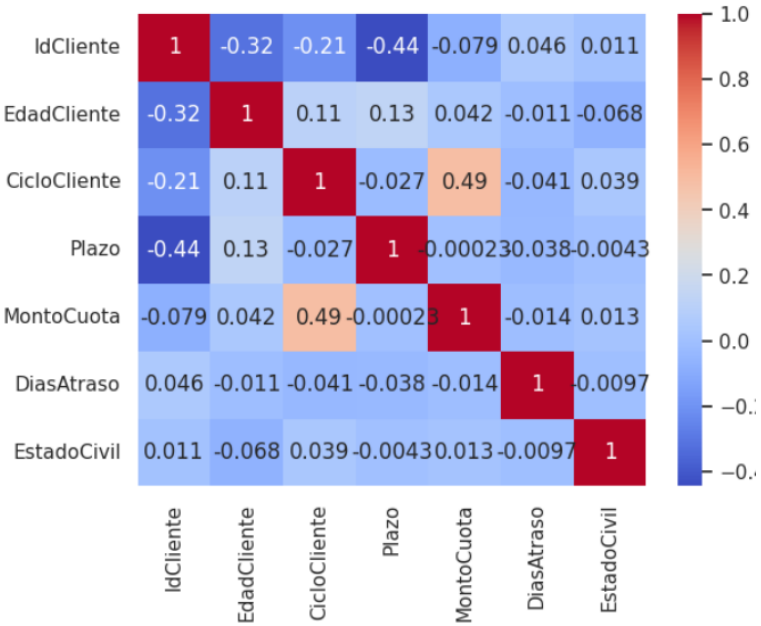


Antes de recortar los valores atípicos *Después del recorte de valores atípicos*

En el siguiente gráfico se puede apreciar un incremento tanto en los montos de las cuotas como en los saldos de capital a partir del 2018 en adelante, es muy sutil determinar el efecto por pandemia en el 2020 que es donde se existe un incremento de saldos pendientes de pago que son los castigos.



Posteriormente se procede a realizar una matriz de correlación para identificar las variables que puedan tener correlaciones altas, logrando observar una correlación muy baja entre las variables montoCuota y CicloCliente, todas las demás variable tienen una correlación muy baja o nula, en general no tienen correlaciones.

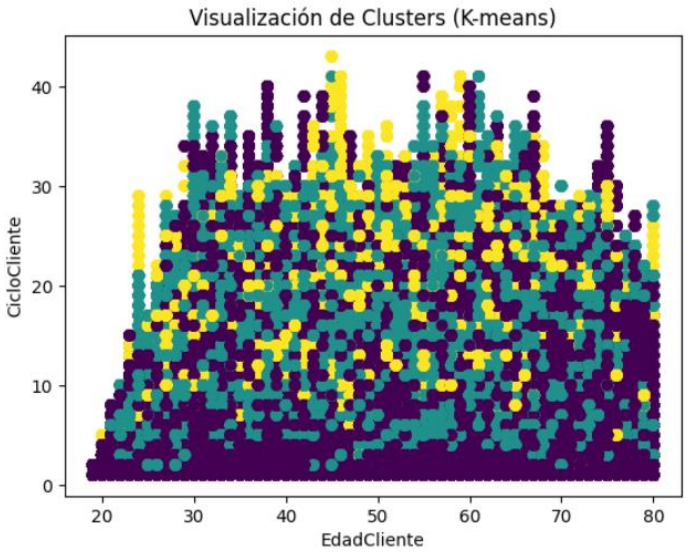


Posteriormente se logra observar la distribución de las variables IdCliente, EdadCliente, CicloCliente, Plazo, MontoCuota, DiasAtraso y EstadoCivil a atraves de los percentiles 10, 25, 50, 75 y 90

```
newgpdf.quantile([.1, .25, .5, .75, .90], axis=0, numeric_only=True)
```

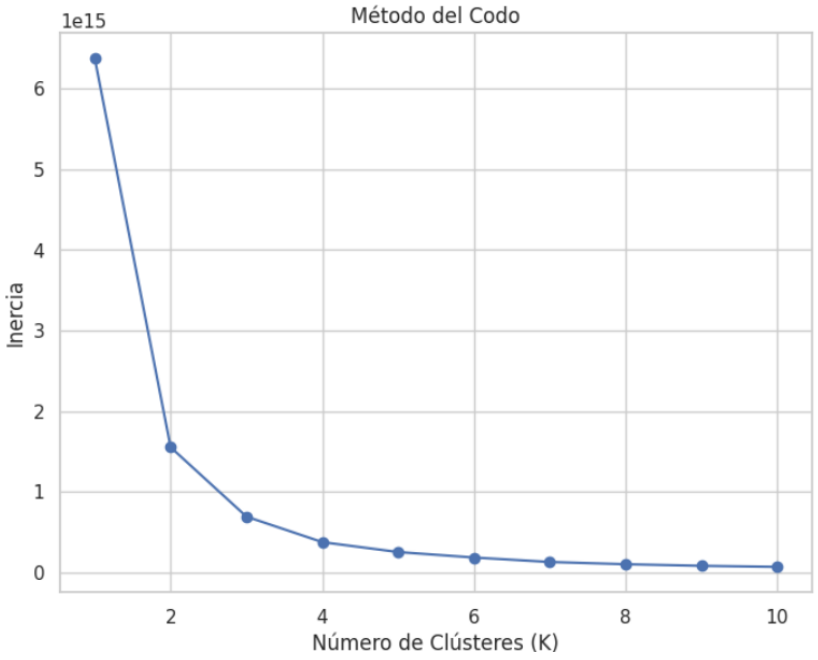
	IdCliente	EdadCliente	CicloCliente	Plazo	MontoCuota	DiasAtraso	EstadoCivil
0.10	8967.0	30.0	1.0	8.0	243.240	0.0	1.0
0.25	23874.0	35.0	2.0	12.0	370.550	0.0	2.0
0.50	55686.0	45.0	3.0	12.0	565.280	0.0	3.0
0.75	92530.0	55.0	7.0	12.0	968.525	1.0	5.0
0.90	113457.0	64.0	12.0	15.0	1599.550	3.0	5.0

Se puede notar que los datos se agrupan basándose del Ciclo del Cliente y la Edad del cliente por lo que sí existen patrones de agrupación. Para determinar el número óptimo de grupos, se tiene que seleccionar el valor de k en el "codo" del siguiente gráfico, es decir, el punto después del cual la distorsión/inercia comienza a disminuir de forma lineal. Por lo tanto, para los datos otorgados se concluye que el número óptimo de conglomerados para los datos es 3.



A pesar de que se cuenta con más de 4 millones de registros en los datos obtenidos para el análisis exploratorio (EDA), también se puede considerar variables

calculadas como el máximo día de atraso del cliente, días de inactividad del cliente, actividad económica entre otros. Como bien se menciona, el proceso de preparación de datos es crucial en el proceso de aprendizaje automático, ya que garantiza que los datos estén en un formato adecuado para entrenar un modelo.



Agregamos valores máximos, la media y la mediana de la variable DiasAtraso por cliente y por crédito para obtener una muestra concreta sobre el historial de créditos por cliente.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 380284 entries, 0 to 380283
Data columns (total 11 columns):
 #   Column          Non-Null Count   Dtype
---  ------          --------------   -----
 0   IdCliente       380284 non-null  int64
 1   EdadCliente     380284 non-null  int64
 2   NumeroCredito   380284 non-null  int64
 3   PeriodoPago     380284 non-null  object
 4   CicloCliente    380284 non-null  int64
 5   Plazo           380284 non-null  int64
 6   EstadoCivil     380284 non-null  int64
 7   EstatusCredito  380284 non-null  object
 8   DAMax           380284 non-null  float64
 9   DAMean          380284 non-null  float64
 10  DAMedian        380284 non-null  float64
dtypes: float64(3), int64(6), object(2)
memory usage: 31.9+ MB
```

Posteriormente seleccionamos los algoritmos SVM (Support Vector Machine) y RFC (Random Forest Classifier) para desarrollar modelos de clasificación que evalúen el riesgo crediticio de los clientes. El objetivo es predecir la probabilidad de que un cliente pague su crédito, crucial para mitigar riesgos financieros. SVM es adecuado para este problema debido a su capacidad de manejar conjuntos de datos complejos y no linealmente separables, optimizando la separación la separación entre clientes que pagan y los que no, maximizando el margen y minimizando errores de clasificación. Por su parte, RFC es eficaz en la gestión de características no lineales y en la clasificación con grandes conjuntos de datos, capturando relaciones complejas entre las características y la clase objetivo, mejorando la precisión del modelo en la predicción del comportamiento de pago del cliente.

Random Forest Classifier
Accuracy: 0.9981987193815165
Classification Report:

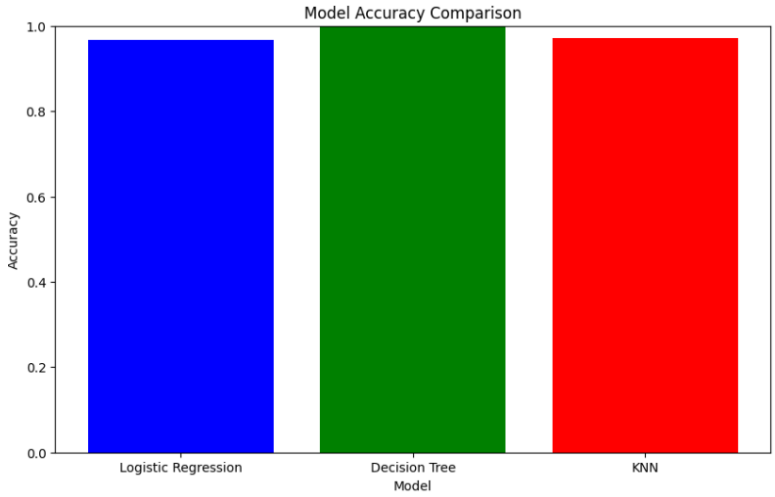
	precision	recall	f1-score	support
CAI	0.98	0.97	0.97	2462
CAN	1.00	1.00	1.00	73595
accuracy			1.00	76057
macro avg	0.99	0.98	0.99	76057
weighted avg	1.00	1.00	1.00	76057

Support Vector Machine Classifier
Accuracy: 0.9676295410021432
Classification Report:

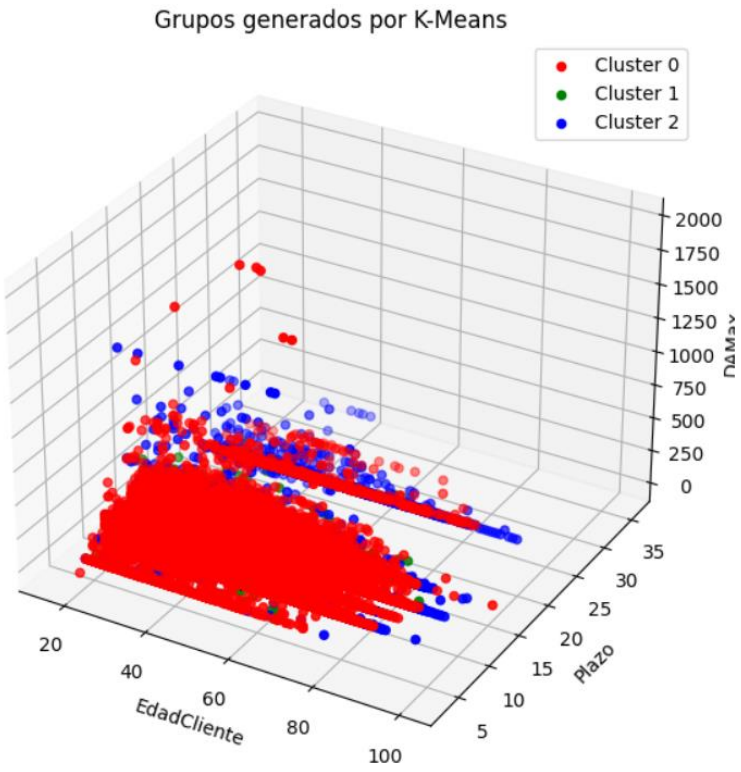
	precision	recall	f1-score	support
CAI	0.00	0.00	0.00	2462
CAN	0.97	1.00	0.98	73595
accuracy			0.97	76057
macro avg	0.48	0.50	0.49	76057
weighted avg	0.94	0.97	0.95	76057

Además de los modelos previamente mencionados, se decide utilizar los algoritmos de Regresión Logística, Árbol de Decisión y KNN (K-Nearest Neighbors) para desarrollar modelos de clasificación destinados a evaluar el riesgo crediticio de los clientes, con el objetivo de de predecir si un cliente pagará su crédito. La Regresión Logística es una opción sólida como baseline para nuestro conjunto de datos de tamaño moderado, al igual que Los Árboles de Decisión. El clasificador KNN ofrece otra perspectiva como baseline en la clasificación. Utilizamos la columna “EstatusCredito” como nuestra variable objetivo, donde “CAN” indica un pago completo del crédito y “CAI” indica un crédito castigado por no pagar.

```
Logistic Regression: 0.9676295410021432
Decision Tree: 0.9966998435383988
KNN: 0.9707456249917825
```



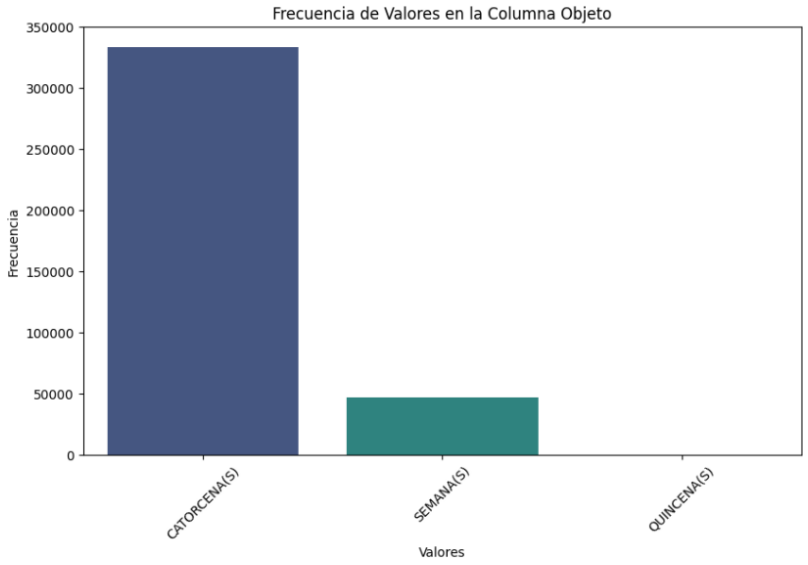
Adicionalmente, se emplea k-means para segmentar a los clientes en tres grupos, ajustando así el sentimiento en el chatbot y proporcionando una comprensión más profunda de las preferencias y comportamientos de los clientes para una interacción más efectiva.



Modelos generados y razones de la elección del modelo final

Realizamos una evaluación sobre las variables a utilizar para entrenar al menos seis modelos, evaluar las métricas durante el proceso de validación cruzada, aplicando una transformación logarítmica y un escalado Min-Max a las columnas numéricas, utilizando One-hot a las columnas categóricas.

Segmentamos los datos utilizando los de mayor frecuencia en la variable PeriodoPago, ya que el comportamiento de un crédito de pago CATORCENA(S) puede ser diferente uno de pago SEMANA(S) o QUINCENA(S).




```
# filtramos solo los creditos CATORCENA(S)
newdf = newgpdf[newgpdf['PeriodoPago']=='CATORCENA(S)'][columnas]
```

Convertimos la columna EstadoCredito en valores numéricos, donde “CAI” es “0” y “CAN” es “1”, mapeamos y dividimos los datos en conjunto de entrenamiento del 80% y prueba del 20%.

```
Dimensiones del conjunto de entrenamiento (X): (266674, 8)
Dimensiones del conjunto de prueba (X): (66669, 8)
Dimensiones del conjunto de entrenamiento (y): (266674,)
Dimensiones del conjunto de prueba (y): (66669,)
```

Aplicamos la transformación logarítmica y el escalado a las columnas numéricas, aplicamos la codificación One-Hot a las columnas categóricas y se dejan algunas columnas sin transformación y combinamos todas las transformaciones en un solo transformador. Juntamos los conjuntos de entrenamiento y prueba en un solo conjunto para realizar validación cruzada. Construimos seis modelos diferentes, utilizando diferentes algoritmos.

Se evalúan las siguientes métricas durante el proceso de validación cruzada:

- **Training time:** El tiempo total que tomó entrenar el modelo.
- **test_f1 y train_f1:** El valor F1, que es la media armónica de la precisión y el recall, tanto en el conjunto de prueba como en el conjunto de entrenamiento.
- **test_roc_auc y train_roc_auc:** El área bajo la curva ROC, que mide la capacidad del modelo para distinguir entre las clases.
- **test_precision y train_precision:** La precisión, que es el número de verdaderos positivos dividido por el número de verdaderos positivos más los falsos positivos.
- **test_accuracy y train_accuracy:** La precisión global del modelo, que es el número de predicciones correctas dividido por el número total de predicciones.
- **test_recall y train_recall:** El recall, que es el número de verdaderos positivos dividido por el número de verdaderos positivos más los falsos negativos.

Obtuvimos los siguientes resultados:

```
>> LR
        Training time: 30.171 seconds
        test_precision 0.968 (0.000)
        train_precision 0.968 (0.000)
        test_f1 0.984 (0.000)
        train_f1 0.984 (0.000)
        test_recall 1.000 (0.000)
        train_recall 1.000 (0.000)
        test_accuracy 0.968 (0.000)
        train_accuracy 0.968 (0.000)
        test_roc_auc 0.330 (0.006)
        train_roc_auc 0.330 (0.002)
```

```
>> kNN
        Training time: 162.876 seconds
        test_precision 0.987 (0.000)
        train_precision 1.000 (0.000)
        test_f1 0.992 (0.000)
        train_f1 1.000 (0.000)
        test_recall 0.997 (0.000)
        train_recall 1.000 (0.000)
        test_accuracy 0.984 (0.001)
        train_accuracy 1.000 (0.000)
        test_roc_auc 0.865 (0.005)
        train_roc_auc 1.000 (0.000)
```

```
>> DTree
        Training time: 23.107 seconds
        test_precision 0.998 (0.000)
        train_precision 0.999 (0.000)
        test_f1 0.998 (0.000)
        train_f1 0.998 (0.000)
        test_recall 0.998 (0.000)
        train_recall 0.998 (0.000)
        test_accuracy 0.996 (0.000)
        train_accuracy 0.996 (0.000)
        test_roc_auc 0.993 (0.001)
        train_roc_auc 0.994 (0.000)
```


```
>> DTree
        Training time: 23.107 seconds
        test_precision 0.998 (0.000)
        train_precision 0.999 (0.000)
        test_f1 0.998 (0.000)
        train_f1 0.998 (0.000)
        test_recall 0.998 (0.000)
        train_recall 0.998 (0.000)
        test_accuracy 0.996 (0.000)
        train_accuracy 0.996 (0.000)
        test_roc_auc 0.993 (0.001)
        train_roc_auc 0.994 (0.000)
```


```
>> MLP
        Training time: 2822.406 seconds
        test_precision 0.970 (0.008)
        train_precision 0.970 (0.008)
        test_f1 0.918 (0.245)
        train_f1 0.918 (0.245)
        test_recall 0.933 (0.249)
        train_recall 0.933 (0.249)
        test_accuracy 0.906 (0.233)
        train_accuracy 0.906 (0.233)
        test_roc_auc 0.500 (0.000)
        train_roc_auc 0.500 (0.000)
```

```
>> RandomForest
        Training time: 819.097 seconds
        test_precision 0.999 (0.000)
        train_precision 1.000 (0.000)
        test_f1 0.999 (0.000)
        train_f1 1.000 (0.000)
        test_recall 0.999 (0.000)
        train_recall 1.000 (0.000)
        test_accuracy 0.998 (0.000)
        train_accuracy 1.000 (0.000)
        test_roc_auc 0.989 (0.002)
        train roc auc 1.000 (0.000)
```

- **Regresión Logística (LR):** La métrica de ROC AUC es extremadamente baja (0.330), lo que indica que el modelo tiene un bajo rendimiento en distinguir entre las clases. La precisión, el recall y la exactitud son altas, pero esto podría deberse a un desequilibrio de clases en el conjunto de datos.
- **k-Vecinos más Cercanos (kNN):** Las métricas en el conjunto de entrenamiento son perfectas (1.000), lo que sugiere sobreajuste. Las métricas en el conjunto de prueba también son altas, especialmente el ROC AUC (0.865), lo cual es una buena señal, aunque podría beneficiarse de una mayor generalización.
- **Árbol de Decisiones (DTree):** Las métricas son muy altas tanto en el conjunto de entrenamiento como en el de prueba, con un ligero indicio de sobreajuste. El tiempo de entrenamiento es razonablemente corto.
- **XGBoost:** Muy buenas métricas tanto en el conjunto de entrenamiento como en el de prueba, pero puede haber un ligero sobreajuste. Buen rendimiento general, con tiempos de entrenamiento aceptables.
- **Perceptrón Multicapa (MLP):** El ROC AUC es 0.500, lo cual indica que el modelo no tiene capacidad de discriminación entre clases. Alto tiempo de

entrenamiento, pero las métricas de precisión, recall y exactitud son similares a las de la regresión logística, lo cual no justifica el tiempo de entrenamiento.

- **Random Forest:** Las métricas son excelentes para ambos conjuntos, lo que sugiere un modelo bien ajustado con buena generalización. El roc_auc alto (0.989) indica una excelente capacidad de discriminación entre clases.

Realizamos un ajuste utilizando RandomizedSearchCV para explorar el espacio de hiperparámetros mediante una estrategia de muestreo aleatorio, lo que resulta más eficiente en términos de tiempo computación que GridSearchCV.

Definimos los espacio de búsqueda de hiperparámetros para cada modelo y ajustamos cada uno de ellos utilizando RandomizedSearchCV durante el proceso de entrenamiento.

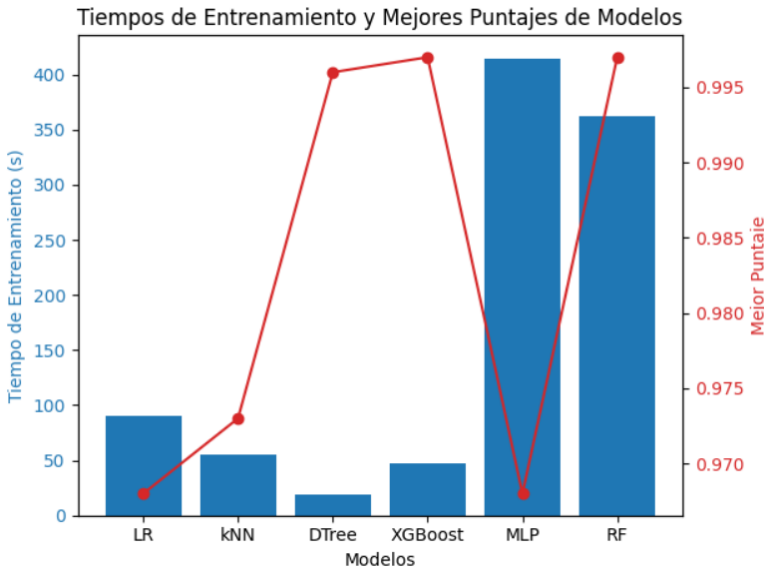
- **param_grid:** Define el espacio de búsqueda de hiperparámetros para cada modelo. Por ejemplo, para la regresión logística, se varían C y solver.
- **RandomizedSearchCV:** Configura la búsqueda aleatoria sobre el espacio de hiperparámetros definido. n_iter=50 especifica que se harán 50 combinaciones aleatorias de hiperparámetros.
- **random_search.fit(Xtrainval, ytrainval):** Ajusta el modelo utilizando RandomizedSearchCV.
- **random_search.best_estimator_:** Obtiene el mejor estimador (modelo con los mejores hiperparámetros).
- **random_search.best_score_:** Muestra la mejor puntuación obtenida durante la búsqueda.
- **random_search.best_params_:** Muestra los mejores parámetros encontrados durante la búsqueda.

```
>> LR
         Training time: 90.187 seconds
         Best Score: 0.968
         Best Params: {'m__C': 3.7554011884736247, 'm__solver': 'newton-cg'}
>> kNN
         Training time: 55.012 seconds
         Best Score: 0.973
         Best Params: {'m__n_neighbors': 29, 'm__weights': 'distance'}
>> DTree
         Training time: 19.118 seconds
         Best Score: 0.996
         Best Params: {'m__criterion': 'gini', 'm__max_depth': 8}
>> XGBoost
         Training time: 47.181 seconds
         Best Score: 0.997
         Best Params: {'m__learning_rate': 0.12473859738014881, 'm__max_depth': 6, 'm__n_est
imators': 113}
>> MLP
         Training time: 414.841 seconds
         Best Score: 0.968
         Best Params: {'m__activation': 'tanh', 'm__hidden_layer_sizes': (50,), 'm__learning
_rate_init': 0.01934347898661638}
>> RF
         Training time: 362.767 seconds
         Best Score: 0.997
         Best Params: {'m__criterion': 'gini', 'm__max_depth': 16, 'm__n_estimators': 63}
```

Realizamos algunos cambios en las transformaciones de las variables numéricas y categóricas mejoramos los modelos con hiperparametros y obtuvimos los siguientes resultados

- La regresión logística logró un puntaje alto de 0.968 con un tiempo de entrenamiento relativamente corto. Esto indica que el modelo es eficiente y preciso para este conjunto de datos. El parámetro C elevado sugiere que el modelo está dando más importancia a los datos de entrenamiento para mejorar el ajuste.
- El modelo kNN obtuvo un excelente puntaje de 0.973, y el tiempo de entrenamiento fue relativamente corto. El uso de un alto número de vecinos (n_neighbors: 29) y la ponderación por distancia (weights: 'distance') ayudaron a mejorar la precisión del modelo.
- El árbol de decisión obtuvo un puntaje casi perfecto de 0.996 con un tiempo de entrenamiento muy bajo. La profundidad máxima de 8 (max_depth: 8) muestra que el modelo no es excesivamente complejo, lo cual es beneficioso para evitar el sobreajuste.

- XGBoost logró el puntaje más alto de 0.997 con un tiempo de entrenamiento razonable. Los parámetros indican un buen equilibrio entre la profundidad del árbol (max_depth: 6), la tasa de aprendizaje (learning_rate: 0.1247) y el número de estimadores (n_estimators: 113), optimizando el rendimiento sin caer en el sobreajuste.
- El modelo MLP alcanzó un puntaje de 0.968 con un tiempo de entrenamiento considerablemente largo. La configuración de una sola capa oculta con 50 neuronas y la función de activación tanh indican una arquitectura simple pero eficaz, aunque el tiempo de entrenamiento sugiere que el modelo es más costoso en términos computacionales.
- El bosque aleatorio también obtuvo un puntaje excelente de 0.997, con un tiempo de entrenamiento considerable. Los parámetros indican un modelo complejo con una profundidad máxima de 16 (max_depth: 16) y un número razonable de estimadores (n_estimators: 63), proporcionando un equilibrio entre precisión y generalización.



Estos resultados subrayan la alta precisión de los modelos XGBoost y RandomForest, mientras que Decision Tree se destaca por su eficiencia en el tiempo de entrenamiento. No obstante, el MLP presenta un costo computacional

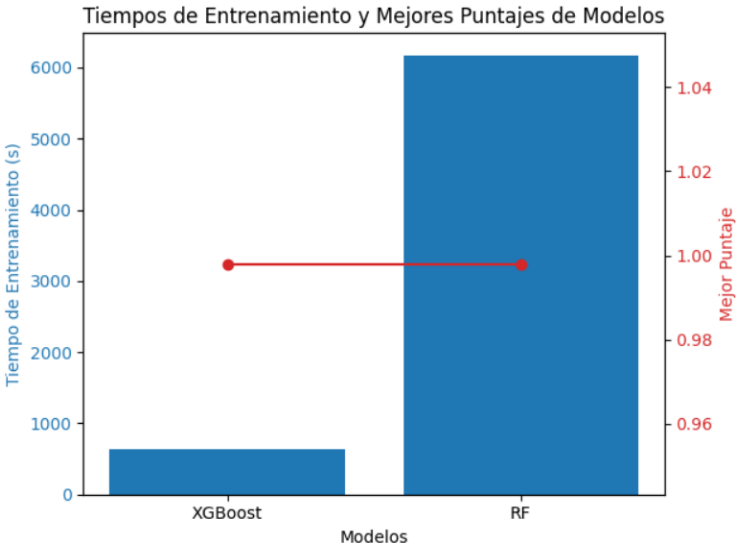
considerablemente más alto, lo cual es un factor crucial a considerar al seleccionar un modelo para aplicaciones en tiempo real o con recursos limitados.

Según lo resultados obtenidos y la gráfica correspondiente, los modelos con mejor rendimiento en términos de “Best Score” son XGBoost y Random Forest, ambos alcanzando un “Best Score” de 0.997, Sin embargo, el modelo XGBoost destaca por su menor tiempo de entrenamiento (47.181 segundos) en comparación con Random Forest (362.767 segundos). Este factor puede ser crucial dependiendo del contexto y los recursos disponibles para el entrenamiento del modelo.

```
>> XGBoost
        Training time: 638.771 seconds
        Best Score: 0.998
        Best Params: {'m__learning_rate': 0.22818159875692626, 'm__max_depth': 9, 'm__n_est
imators': 89}
>> RF
        Training time: 6174.234 seconds
        Best Score: 0.998
        Best Params: {'m__criterion': 'gini', 'm__max_depth': 17, 'm__n_estimators': 156}
```

Los modelos ajustados de XGBoost y Random Forest demuestran un rendimiento sobresaliente, ambos alcanzando un “Best Score” de 0.998. XGBoost destaca por un tiempo de entrenamiento razonable de aproximadamente 639 segundo (algo más de 10 minutos) y una precisión cercana al 100%. Este balance entre tiempo de entrenamiento y rendimiento es notable. Los parámetros óptimos incluyen un learning_rate relativamente alto (0.228), una profundidad máxima del árbol (max_depth) de 9 y un número moderado de estimadores (n_estimators) de 89. Estos parámetros sugieren un modelo complejo pero no excesivamente profundo, ayudando a evitar el sobreajuste mientras se captura la complejidad del problema.

El entrenamiento de Random Forest toma considerablemente más tiempo, aproximadamente 6174 segundos (algo más de 1 hora y 43 minutos). Aunque logra el mismo rendimiento que XGBoost, su costo computacional es significativamente mayor. Los parámetros óptimos incluyen una moyor profundidad máxima del árbol (max_depth) de 17 y un número elevado de estimadores (n_estimators) de 156. Esto indica que Random Forest utiliza numerosos árboles profundos para captar la complejidad del problema, lo cual explica el extenso tiempo de entrenamiento.



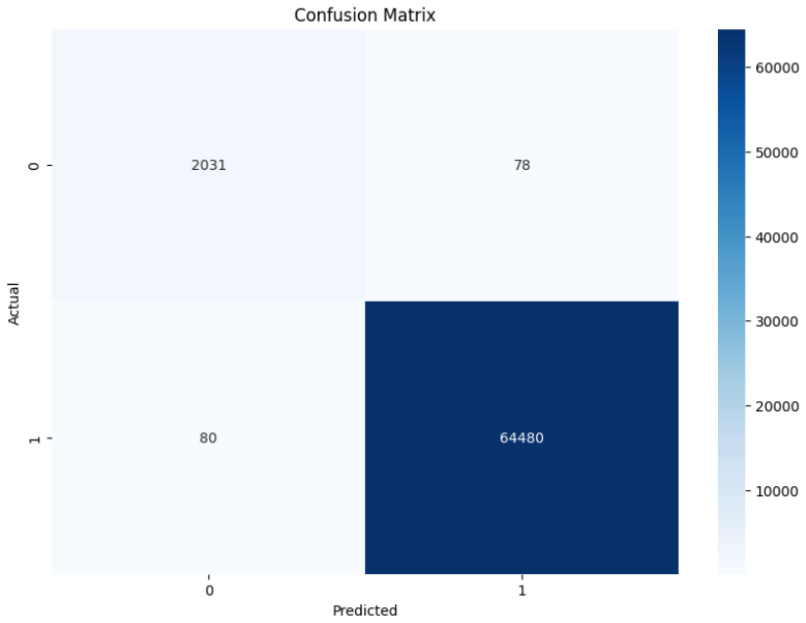
Ambos modelos lograron un rendimiento casi perfecto con un “Best Score” de 0.998, lo cual es excelente. Sin embargo, XGBost es mucho más eficiente en términos de tiempo de entrenamiento en comparación con Random Forest. Dado que XGBost requiere menos tiempo de computación, es preferible en situaciones donde la rapidez es crucial. Por otro lado, si no hay limitaciones de tiempo y recursos computacionales, Random Forest sigue siendo una opción robusta debido a su capacidad para manejar grandes volúmenes de datos y su resistencia al sobreajuste.

Para problemas de clasificación o regresión con conjuntos de datos moderados a grandes, técnicas de ensamble como Random Forest (Bagging) o Grandient Boosting (Boosting) son recomendables. Ambas son robustas, manejan bien diferentes tipos de datos y son fáciles de implementar.

Para conjuntos de datos con variables numéricas y categóricas, Random Forest es una opción sólida debido a su capacidad para manejar diversos tipos de datos y su resistencia al sobreajuste.

```
Precisión del modelo Random Forest: 0.9976300829470969
```

La precisión del modelo Random Forest es un resultado extremadamente alto. Sin embargo, es importante interpretar esta métrica con cuidado y considerar otros factores antes de concluir que el modelo es excelente.



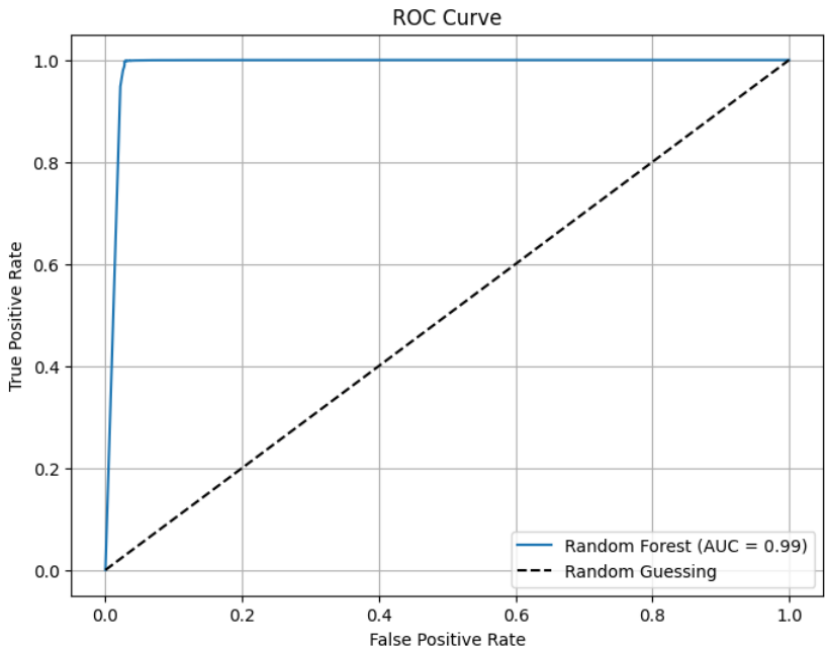
Los resultados de la matriz de confusión indican que el modelo Random Forest está funcionando de manera excepcionalmente buena.

Reporte de Clasificación:

	precision	recall	f1-score	support
0	0.96	0.96	0.96	2109
1	1.00	1.00	1.00	64560
accuracy			1.00	66669
macro avg	0.98	0.98	0.98	66669
weighted avg	1.00	1.00	1.00	66669

El modelo muestra un rendimiento perfecto para la clase mayoritaria (clase 1), con una precisión, recall y F1-score de 1.00, lo que indica su efectividad en identificar correctamente estos casos. Aunque el soporte para la clase minoritaria (clase 0) es considerablemente menor, el modelo mantiene una alta precisión, recall y F1-score de 0.96, demostrando su capacidad para manejar bien las clases minoritarias. Una exactitud del 100% es notable, pero es crucial verificar que el modelo no esté sobre ajustado ni presente sesgos en los datos.

La métrica de promedio macro indica un buen equilibrio en el rendimiento del modelo entre las clases, a pesar del desequilibrio en el soporte. En aplicaciones prácticas, un modelo con estas métricas sería muy confiable. Sin embargo, siempre es aconsejable realizar validaciones adicionales con datos nuevos para asegurar la robustez del modelo.



La curva ROC del modelo, con un AUC de 0.988, sugiere un desempeño excelente. Esto indica que el modelo es muy eficaz en la discriminación entre clases positivas y negativas, con una alta sensibilidad y especificidad a lo largo de diferentes umbrales de decisión. Este resultado refuerza la confianza del modelo y su aplicabilidad en contextos reales, ya que ofrece una combinación óptima de precisión y robustez.

Recomendaciones clave para la implementar la solución

El rendimiento del modelo es suficientemente bueno para su implementación en producción. Tanto XGBoost como Random Forest muestran un rendimiento extremadamente alto con un “Best Score” de 0.998, lo que indica una precisión casi perfecta. XGBoost destaca por su eficiencia con el tiempo de entrenamiento de

aproximadamente 639 segundos, lo que lo hace ideal para entornos donde el tiempo de entrenamiento es crucial. Por otro lado, aunque Random Forest requiere más tiempo de entrenamiento, sigue siendo una opción robusta y eficaz, especialmente para conjuntos de datos con variables numéricas y categóricas.

Además, la precisión del modelo Random Forest de 0.9976, junto con una matriz de confusión que muestran una alta tasa de aciertos y un AUC de 0.988, sugiere un desempeño excelente en la discriminación entre clases. Estos resultados indican que el modelo es preciso, robusto y confiable para aplicaciones reales. Sin embargo, es esencial seguir monitoreando el rendimiento del modelo en producción y considerar factores como la capacidad de generalización y el costo computacional, dependiendo del contexto y los recursos disponibles.

Para confirmar la robustez del modelo seleccionado y garantizar que generalice bien a nuevas muestras, es crucial realizar validaciones adicionales en datos no vistos, lo que ayudará a identificar y evitar posibles problemas de sobreajuste.

Utilizar técnicas de búsqueda exhaustiva, como Grid Search, para optimizar los hiperparámetros y maximizar el rendimiento del modelo. Implementar un sistema de monitoreo continuo para evaluar el rendimiento del modelo en producción, detectando desviaciones o degradaciones en su desempeño de manera oportuna.

Evaluar la escalabilidad del modelo para asegurar que pueda manejar mayores volúmenes de datos en producción. Verificar que la infraestructura y los recursos computacionales disponibles sean adecuados para soportar la carta de trabajo esperada. Documentar exhaustivamente el proceso de desarrollo del modelo, incluyendo los pasos de preprocesamiento de datos, selección de características, ajuste de hiperparámetros y evaluación del modelo. Establecer un plan de mantenimiento para actualizar y mejorar el modelo con el tiempo, asegurando su relevancia y efectividad continua.

Integrar el modelo en el flujo de trabajo existente de la organización, garantizando su compatibilidad con otras herramientas y sistemas utilizados. Definir claramente

los stakeholders y asegurar su participación en la implementación del modelo, facilitando una comunicación efectiva y colaboración durante todas las etapas del proyecto. Proporcionar capacitación adecuada al personal encargado de operar y mantener la interpretación de los resultados. Establecer un proceso de gestión de cambios para realizar actualizaciones y mejoras en el modelo de manera controlada, minimizando el impacto en la producción y asegurando una transición suave.

A continuación se presentan las tareas y procedimientos accionables para los stakeholders internos y externos de la organización:

Equipo de desarrollo de modelos:

- Desarrollar el modelo Random Forest conforme a los requisitos y estándares establecidos.
- Realizar pruebas exhaustivas para asegurar la funcionalidad y precisión del modelo.
- Documentar todo el proceso de desarrollo, proporcionando información técnica detallada para su revisión.

Equipo de TI / Ingeniería:

- Implementar el modelo en el entorno de producción de manera segura y eficiente.
- Configurar sistemas de monitoreo y alertas para detectar problemas en tiempo real.
- Asegurar la integración adecuada del modelo con otros sistemas y procesos existentes.

Equipo de operaciones:

- Coordinar con el equipo de TI para garantizar una implementación fluida del modelo en producción.
- Establecer protocolos de escalación para manejar problemas o emergencias relacionadas con el modelo.

- Proporcionar capacitación y soporte continuo a los usuarios finales que interactúan con el modelo.

Equipo de gestión / directivos:

- Revisar y aprobar el plan de implementación del modelo, asignando los recursos necesarios para su ejecución.
- Supervisar el progreso de la implementación y tomar decisiones estratégicas basadas en los resultados del modelo.
- Comunicar los beneficios y la importancia del modelo a otras partes interesadas dentro de la organización.

Clientes / usuarios finales:

- Proporcionar retroalimentación sobre la usabilidad y efectividad del modelo una vez implementado.
- Comunicar cualquier problema o dificultad encontrada al interactuar con el modelo en producción.
- Utilizar el modelo de manera efectiva para tomar decisiones informadas y mejorar los resultados comerciales.

Socios comerciales:

- Colaborar con la organización para integrar el modelo en procesos comerciales compartidos.
- Proporcionar datos adicionales o insights relevantes que puedan mejorar el rendimiento del modelo.
- Participar en discusiones estratégicas sobre cómo aprovechar mejor los resultados del modelo para beneficio mutuo.

Reguladores / entidades de cúmplimiento:

- Asegurarse de que la implementación del modelo cumpla con todas las regulaciones y normativas aplicables.

- Proporcionar orientación y supervisión sobre el uso ético y legal de los datos y las predicciones del modelo.
- Participar en auditorías y revisiones periódicas para garantizar la transparencia y responsabilidad en el uso del modelo.

Al asignar roles y responsabilidades específicos a cada grupo de stakeholders, se puede asegurar una implementación efectiva y exitosa del modelo Random Forest, maximizando su valor y beneficio tanto para la organización como para sus socios y clientes.

Análisis costo beneficio

A continuación se presentan los costos asociados al proyecto y se plantea una solución en la plataforma AWS, utilizando el servicio de Amazon Sage Maker para ejecutar el modelo en la nube.

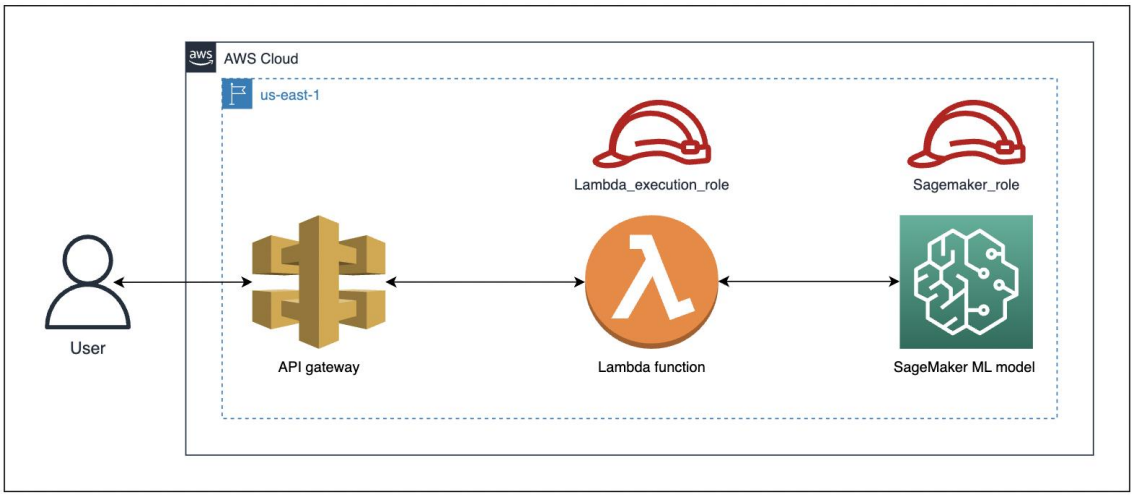
Durante las etapas de Comprensión del Negocio, Adquisición de Datos, Preparación de Datos, Modelado, y Evaluación no se tuvieron gastos. Para la implementación se evaluaron diferentes plataformas para ejecutar, y entrenar modelos de machine learning. Los costos por hora dependiendo de la memoria y CPU:

Region: US East (Ohio)

Standard Instances	vCPU	Memory	Price per Hour
ml.t3.medium	2	4 GiB	$0.05
ml.t3.large	2	8 GiB	$0.10
ml.t3.xlarge	4	16 GiB	$0.20
ml.t3.2xlarge	8	32 GiB	$0.399
ml.m7i.large	2	8 GiB	$0.121
ml.m7i.xlarge	4	16 GiB	$0.242
ml.m7i.2xlarge	8	32 GiB	$0.484
ml.m7i.4xlarge	16	64 GiB	$0.968
ml.m7i.8xlarge	32	128 GiB	$1.935
ml.m7i.12xlarge	48	192 GiB	$2.903
ml.m7i.16xlarge	64	256 GiB	$3.871
ml.m7i.24xlarge	96	384 GiB	$5.806
ml.m7i.48xlarge	192	768 GiB	$11.612
ml.m6i.large	2	8 GiB	$0.115

Amazon Sage Maker es un servicio que proporciona servicios de Machine Learning, el cual ofrece un entorno integrado de Jupyter Notebooks donde se pueden hacer las tareas de acceder a datos, analizar y extraer características, entrenar modelos, evaluarlos y desplegarlos a un entorno alojado en la nube.

En el siguiente diagrama mostramos la arquitectura ideal para un proyecto basado en una jupyter notebook, la cual es guardada en un endpoint y accedida mediante funciones lambdas. Utilizando estas herramientas tendremos facilidad de uso, escalabilidad automática, entrenamiento distribuido y despliegue seguro.



Utilizar AWS Lambda y Amazon SageMaker en conjunto ofrecen beneficios clave para el desarrollo, despliegue y operación de modelos de inteligencia artificial de manera eficiente y escalable. AWS Lambda proporciona un entorno sin servidor que permite ejecutar código de forma automatizada y bajo demanda. Esto se traduce en una reducción significativa de la carga operativa, ya que Lambda escala automáticamente para manejar cualquier cantidad de solicitudes entrantes, optimizando los costos al ejecutar código solo cuando sea necesario.

Por otro lado, Amazon SageMaker simplifica todo el ciclo de vida del aprendizaje automático, desde la preparación y el procesamiento de datos hasta el entrenamiento y el despliegue de modelos.

Riesgos y Desafíos

La implementación de un modelo Random Forest para evaluar el riesgo crediticio, aunque prometedora, conlleva varios riesgos y desafíos potenciales.

Sobreajuste (Overfitting): El modelo puede funcionar extremadamente bien en los datos de entrenamiento, pero no generalizar adecuadamente a nuevos datos. Esto puede llevar a una precisión aparente que no se mantiene en producción.

Sesgo en los Datos: Si los datos de entrenamiento tienen sesgos, el modelo también los aprenderá, lo que puede resultar en decisiones injustas o discriminatorias. Es crucial asegurarse de que los datos sean representativos y libres de sesgos.

Complejidad Computacional: Random Forest puede ser intensivo en términos computacionales y memoria, especialmente con grandes conjuntos de datos y numerosos árboles. Esto puede requerir recursos significativos y aumentar los costos operativos.

Mantenimiento Continuo: Los modelos de machine learning requieren monitoreo y actualización constantes para mantener su precisión y relevancia. Cambios en los datos subyacentes pueden requerir retraining frecuente del modelo.

Integración con Sistemas Existentes: Integrar el modelo con los sistemas actuales de la organización puede ser complejo. Asegurar que todos los sistemas se comuniquen correctamente y que los resultados del modelo se utilicen efectivamente puede ser un desafío técnico.

Interpretabilidad del Modelo: Aunque Random Forest es más interpretable que algunos otros modelos, aún puede ser difícil para los stakeholders no técnicos entender las decisiones del modelo. Esto puede dificultar la adopción y confianza en el modelo.

Regulaciones y Cumplimiento: Asegurarse de que el uso del modelo cumpla con todas las leyes y regulaciones aplicables, especialmente en términos de privacidad de datos y equidad, es esencial. No cumplir con las regulaciones puede resultar en sanciones legales.

Dependencia de la Calidad de los Datos: La calidad y cantidad de datos son cruciales para el rendimiento del modelo. Datos incompletos. incorrectos o desactualizados pueden llevar a predicciones inexactas y decisiones incorrectas.

Estabilidad: A medida que el volumen de datos crece, el modelo debe ser capaz de escalar eficientemente. Asegurarse de que la infraestructura soporta este crecimiento es vital para evitar problemas de rendimiento.

Resistencia a Cambios en el Mercado: El modelo debe ser capaz de adaptarse a cambios en el comportamiento del mercado y las condiciones económicas. Un modelo que no se actualiza puede volverse obsoleto rápidamente.

Para mitigar los riesgos se propone.

Validación Cruzada y Pruebas Exhaustivas: Usar técnicas de validación cruzada y mantener un conjunto de datos de pruebas separado para evaluar el rendimiento real del modelo.

Monitoreo y Actualización Continua: Establecer sistemas para el monitoreo continuo del rendimiento del modelo y mecanismos para su actualización regular.

Mejora de la interpretabilidad: Utilizar técnicas de interpretación de modelos puede explicar las decisiones del Random Forest a los stakeholders.

Cumplimiento Regulatorio: Trabajar estrechamente con equipos legales y de cumplimiento para asegurar que todas las leyes y regulaciones se cumplan.

Gestión de Datos: Implementar procesos rigurosos de gestión de datos para asegurar la calidad y representatividad de los datos utilizados para el entrenamiento del modelo.

Bibliografías

Baccarin, F. (2020, diciembre 28). *Machine learning that pays the bills: choosing models in business contexts. Medium*. Reciuperado de: https://towardsdatascience.com/machine-learning-that-pays-the-bills-choosing-models-in-business-contexts-e9003fd434a1

Babic, B., Cohen, I.G., Evgeniou, T., y Gerke, S. (2021). *When Machine Learning Goes Off the Rails*. Harvard Business Review. Recuperado de: https://hbr.org/2021/01/when-machine-learning-goes-off-the-rails

Deloitte Access Economics. (2017). *Business impacts of machine learning*. Deloitte. Recuperado de: https://www2.deloitte.com/content/dam/Deloitte/tr/Documents/process-and-operations/TG_Google%20Machine%20Learning%20report_Digital%20Final.pdf

Baccarin, F. (2020, diciembre 28). Machine learning that pays the bills: choosing models in business contexts. Medium. Recuperado de: https://towardsdatascience.com/machine-learning-that-pays-the-bills-choosing-models-in-business-contexts-e9003fd434a1

Costa, R. (2022). The CRISP-ML Methodology: A Step-by-Step Approach to Real-World Machine Learning Projects. Edición propia.

S. A. (2024, abril 23). El modelo Llama 3 de Meta ya está disponible en Amazon Bedrock, Recuperado de: https://aws.amazon.com/es/about-aws/whats-new/2024/04/meta-llama-3-amazon-bedrock/

Bardoliwalla, N (2024, abril 24) Meta Llama 3 Available Today on Google Cloud Vertex AI. Recuperado de: https://cloud.google.com/blog/products/ai-machine-learning/meta-llama-3-available-today-on-google-cloud-vertex-ai

ThasmikaGokal, (2024, abril 18). Introducing Meta Llama 3 Models on Azure AI Model Catalog. Recuperado de: https://techcommunity.microsoft.com/t5/ai-machine-learning-blog/introducing-meta-llama-3-models-on-azure-ai-model-catalog/ba-p/4117144