1c764da3-c2df-4fcc-8500-5c40b0034f48

February 16, 2025

Hola **Jafet**!

Soy **Patricio Requena** . Es un placer ser el revisor de tu proyecto el día de hoy!

Revisaré tu proyecto detenidamente con el objetivo de ayudarte a mejorar y perfeccionar tus habilidades. Durante mi revisión, identificaré áreas donde puedas hacer mejoras en tu código, señalando específicamente qué y cómo podrías ajustar para optimizar el rendimiento y la claridad de tu proyecto. Además, es importante para mí destacar los aspectos que has manejado excepcionalmente bien. Reconocer tus fortalezas te ayudará a entender qué técnicas y métodos están funcionando a tu favor y cómo puedes aplicarlos en futuras tareas.

Recuerda que al final de este notebook encontrarás un comentario general de mi parte, empecemos!

Encontrarás mis comentarios dentro de cajas verdes, amarillas o rojas, **por favor, no muevas, modifiques o borres mis comentarios** :

Comentario del revisor Si todo está perfecto.

Comentario del revisor Si tu código está bien pero se puede mejorar o hay algún detalle que le hace falta.

Comentario del revisor Si de pronto hace falta algo o existe algún problema con tu código o conclusiones.

Puedes responderme de esta forma:

Respuesta del estudiante

1 Proyecto Sprint 13 "Pronosticos y predicciones"

2 Paso 1. Descargar los datos

3 Cargar Librerias

[81]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
```

```
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score,␣
 ↪confusion_matrix
from scipy.cluster.hierarchy import dendrogram, linkage
from sklearn.cluster import KMeans
```

Comentario del revisor (1ra Iteracion)

Una buena práctica para cuando tengas que importar varias librerías es seguir el siguiente órden en las mismas:

- Primero todas las librerías que vienen ya con python cómo `datetime`, `os`, `json`, etc.
- Luego de las librerías de Python si las de terceros cómo `pandas`, `scipy`, `numpy`, etc.
- Por último, en el caso de que armes tu propio módulo en tu proyecto esto debería ir en tercer lugar, y recuerda siempre ordenar cada tipo por orden alfabético

Para cada sección se deben ordenar en orden alfabético

4 Cargar Datos

[82]:
```
df = pd.read_csv('/datasets/gym_churn_us.csv')
```

5 Paso 2: Análisis Exploratorio de Datos (EDA)

6 Exploracion Inicial

[83]:
```
display(df.head())
display(df.info())
```

```
   gender  Near_Location  Partner  Promo_friends  Phone  Contract_period  \
0       1              1        1              1      0                6
1       0              1        0              0      1               12
2       0              1        1              0      1                1
3       0              1        1              1      1               12
4       1              1        1              1      1                1

   Group_visits  Age  Avg_additional_charges_total  Month_to_end_contract  \
0             1   29                     14.227470                    5.0
1             1   31                    113.202938                   12.0
2             0   28                    129.448479                    1.0
3             1   33                     62.669863                   12.0
4             0   26                    198.362265                    1.0

   Lifetime  Avg_class_frequency_total  Avg_class_frequency_current_month  \
0         3                   0.020398                           0.000000
1         7                   1.922936                           1.910244
```

```
2          2                    1.859098                              1.736502
3          2                    3.205633                              3.357215
4          3                    1.113884                              1.120078

   Churn
0      0
1      0
2      0
3      0
4      0

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4000 entries, 0 to 3999
Data columns (total 14 columns):
 #   Column                             Non-Null Count  Dtype
---  ------                             --------------  -----
 0   gender                             4000 non-null   int64
 1   Near_Location                      4000 non-null   int64
 2   Partner                            4000 non-null   int64
 3   Promo_friends                      4000 non-null   int64
 4   Phone                              4000 non-null   int64
 5   Contract_period                    4000 non-null   int64
 6   Group_visits                       4000 non-null   int64
 7   Age                                4000 non-null   int64
 8   Avg_additional_charges_total       4000 non-null   float64
 9   Month_to_end_contract              4000 non-null   float64
 10  Lifetime                           4000 non-null   int64
 11  Avg_class_frequency_total          4000 non-null   float64
 12  Avg_class_frequency_current_month  4000 non-null   float64
 13  Churn                              4000 non-null   int64
dtypes: float64(4), int64(10)
memory usage: 437.6 KB

None
```

[84]:
```
print("Resumen estadístico:")
display(df.describe())
print("Valores nulos:")
print(df.isnull().sum())
```

```
Resumen estadístico:

            gender  Near_Location      Partner  Promo_friends        Phone  \
count  4000.000000    4000.000000  4000.000000    4000.000000  4000.000000
mean      0.510250       0.845250     0.486750       0.308500     0.903500
std       0.499957       0.361711     0.499887       0.461932     0.295313
min       0.000000       0.000000     0.000000       0.000000     0.000000
25%       0.000000       1.000000     0.000000       0.000000     1.000000
50%       1.000000       1.000000     0.000000       0.000000     1.000000
```

```
75%       1.000000       1.000000     1.000000       1.000000     1.000000
max       1.000000       1.000000     1.000000       1.000000     1.000000

       Contract_period  Group_visits          Age  \
count      4000.000000   4000.000000  4000.000000
mean          4.681250      0.412250    29.184250
std           4.549706      0.492301     3.258367
min           1.000000      0.000000    18.000000
25%           1.000000      0.000000    27.000000
50%           1.000000      0.000000    29.000000
75%           6.000000      1.000000    31.000000
max          12.000000      1.000000    41.000000

       Avg_additional_charges_total  Month_to_end_contract     Lifetime  \
count                   4000.000000            4000.000000  4000.000000
mean                     146.943728               4.322750     3.724750
std                       96.355602               4.191297     3.749267
min                        0.148205               1.000000     0.000000
25%                       68.868830               1.000000     1.000000
50%                      136.220159               1.000000     3.000000
75%                      210.949625               6.000000     5.000000
max                      552.590740              12.000000    31.000000

       Avg_class_frequency_total  Avg_class_frequency_current_month  \
count                4000.000000                        4000.000000
mean                    1.879020                           1.767052
std                     0.972245                           1.052906
min                     0.000000                           0.000000
25%                     1.180875                           0.963003
50%                     1.832768                           1.719574
75%                     2.536078                           2.510336
max                     6.023668                           6.146783

             Churn
count  4000.000000
mean      0.265250
std       0.441521
min       0.000000
25%       0.000000
50%       0.000000
75%       1.000000
max       1.000000

Valores nulos:
gender                             0
Near_Location                      0
Partner                            0
Promo_friends                      0
```

```
Phone                                0
Contract_period                      0
Group_visits                         0
Age                                  0
Avg_additional_charges_total         0
Month_to_end_contract                0
Lifetime                             0
Avg_class_frequency_total            0
Avg_class_frequency_current_month    0
Churn                                0
dtype: int64
```

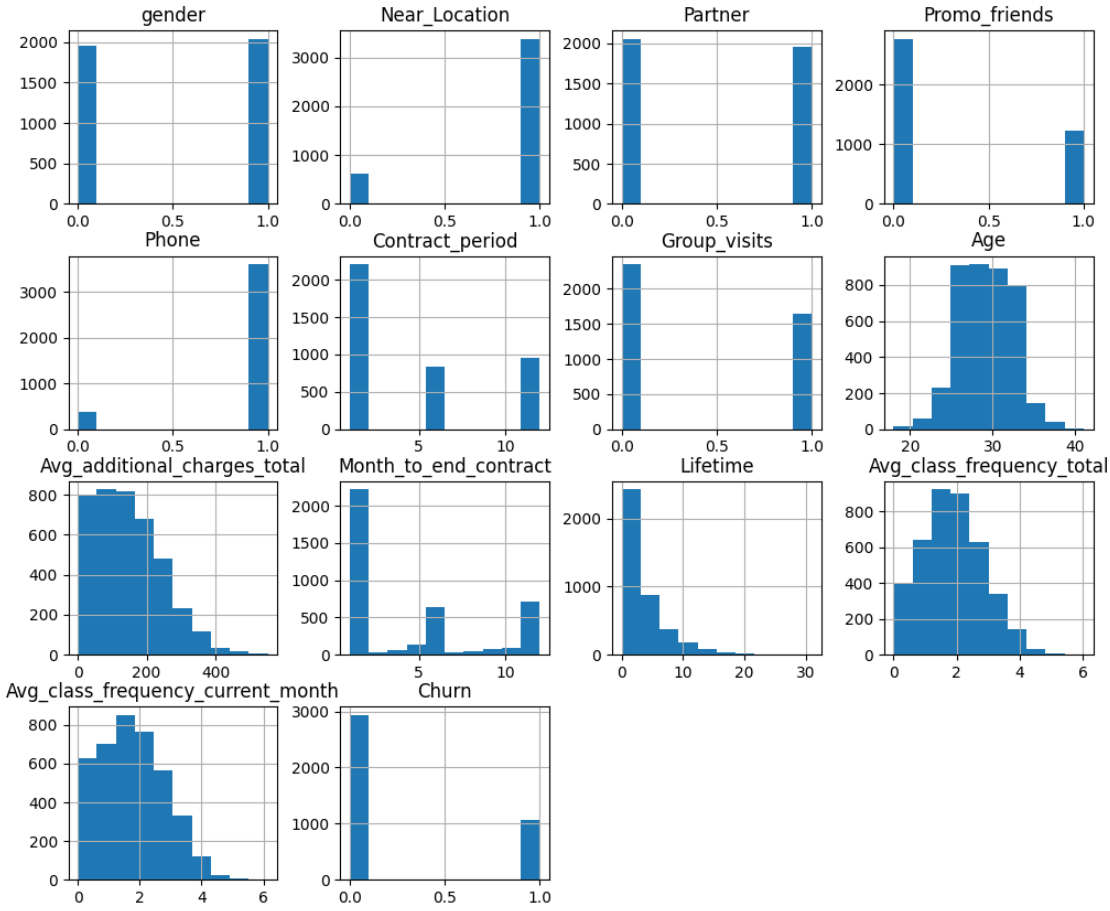
7 Comparación de características entre clientes que se quedaron y los que se fueron

[85]:
```
churn_means = df.groupby("Churn").mean()
display(churn_means)
```

```
         gender  Near_Location   Partner  Promo_friends     Phone  \
Churn
0      0.510037       0.873086  0.534195       0.353522  0.903709
1      0.510839       0.768143  0.355325       0.183789  0.902922

       Contract_period  Group_visits        Age  Avg_additional_charges_total  \
Churn
0             5.747193      0.464103  29.976523                    158.445715
1             1.728558      0.268615  26.989632                    115.082899

       Month_to_end_contract  Lifetime  Avg_class_frequency_total  \
Churn
0                   5.283089  4.711807                   2.024876
1                   1.662582  0.990575                   1.474995

       Avg_class_frequency_current_month
Churn
0                               2.027882
1                               1.044546
```

8 Histogramas de distribución

[86]:
```
df.hist(figsize=(12, 10))
plt.show()
```

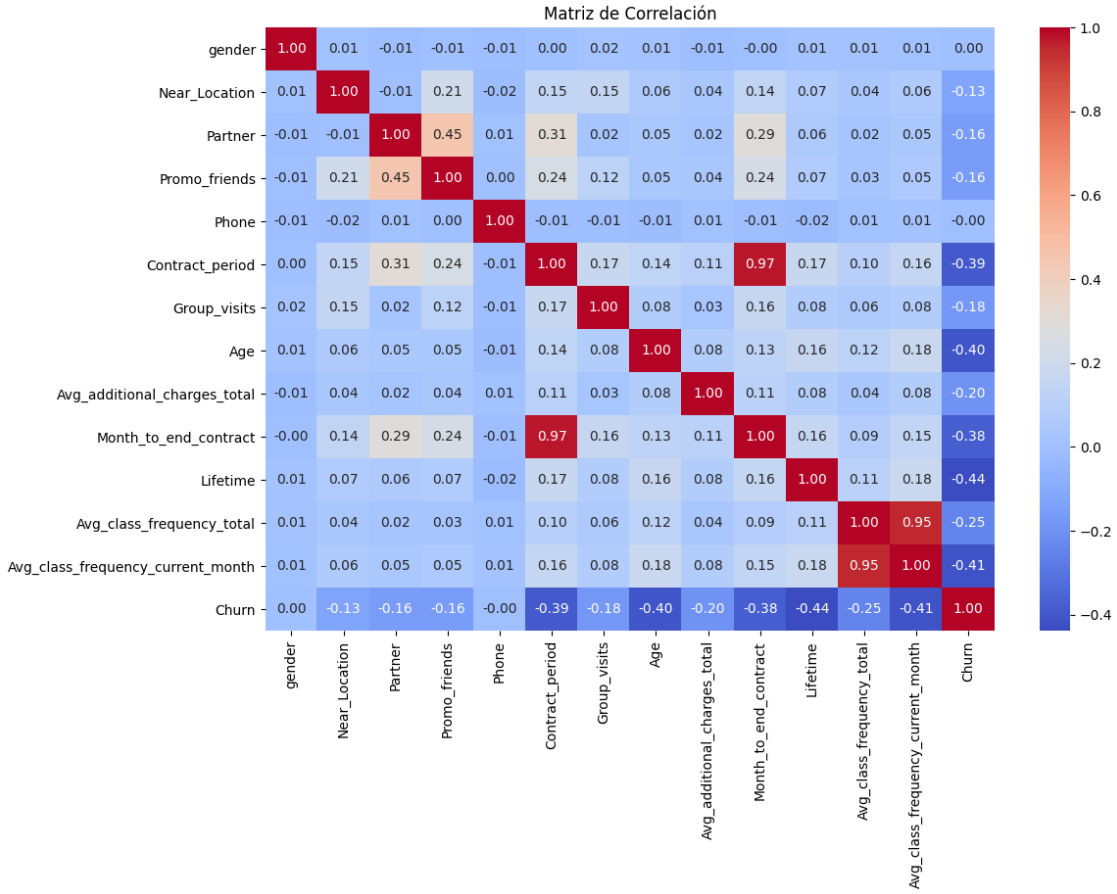


Comentario del revisor (1ra Iteracion)

Muy bien hecho! Estas gráficas son muy claras para mostrar las diferencias entre cada variable de tu dataset, además el redactar conclusiones tan claras de las mismas ayuda mucho a seguir el proceso de análisis

9 Matriz de correlación

[87]:
```
plt.figure(figsize=(12, 8))
sns.heatmap(df.corr(), annot=True, cmap="coolwarm", fmt=".2f")
plt.title("Matriz de Correlación")
plt.show()
```



Comentario del revisor (1ra Iteracion)

Muy bien mostrada la matriz de correlación! Solo ten en cuenta que no siempre correlación significa causalidad, puede que en algunos casos tengas variables altamente correlacionadas pero no necesariamente son causa una de la otra

10 Paso 3: Construcción de Modelo Predictivo

11 Separar características y variable objetivo

[88]:
```
X = df.drop(columns=["Churn"])
y = df["Churn"]
```

12 Dividir en conjunto de entrenamiento y prueba

[89]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
↪random_state=42)
```

13 Estandarizar los datos para mejorar la convergencia del modelo

[90]:
```
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

14 Regresión Logística

[91]:
```
log_reg = LogisticRegression(max_iter=500)  # Se aumenta el número de␣
↪iteraciones
log_reg.fit(X_train_scaled, y_train)
y_pred_log = log_reg.predict(X_test_scaled)
```

15 Bosque Aleatorio

[92]:
```
rf = RandomForestClassifier(random_state=42)
rf.fit(X_train, y_train)
y_pred_rf = rf.predict(X_test)
```

16 Evaluación de Modelos

[93]:
```
def evaluate_model(y_true, y_pred, model_name):
    print(f"{model_name}:")
    print(f"Exactitud: {accuracy_score(y_true, y_pred):.2f}")
    print(f"Precisión: {precision_score(y_true, y_pred):.2f}")
    print(f"Recall: {recall_score(y_true, y_pred):.2f}")
    print(confusion_matrix(y_true, y_pred))
    print("\n")

evaluate_model(y_test, y_pred_log, "Regresión Logística")
evaluate_model(y_test, y_pred_rf, "Bosque Aleatorio")
```

```
Regresión Logística:
Exactitud: 0.92
Precisión: 0.87
Recall: 0.78
[[575  23]
```

```
 [ 44 158]]


Bosque Aleatorio:
Exactitud: 0.91
Precisión: 0.85
Recall: 0.78
[[570  28]
 [ 44 158]]
```

Comentario del revisor (1ra Iteracion)

Muy buen trabajo! Obtuviste resultados muy buenos en el entrenamiento de tus modelos y realizaste el análisis adecuado a partir de los mismos

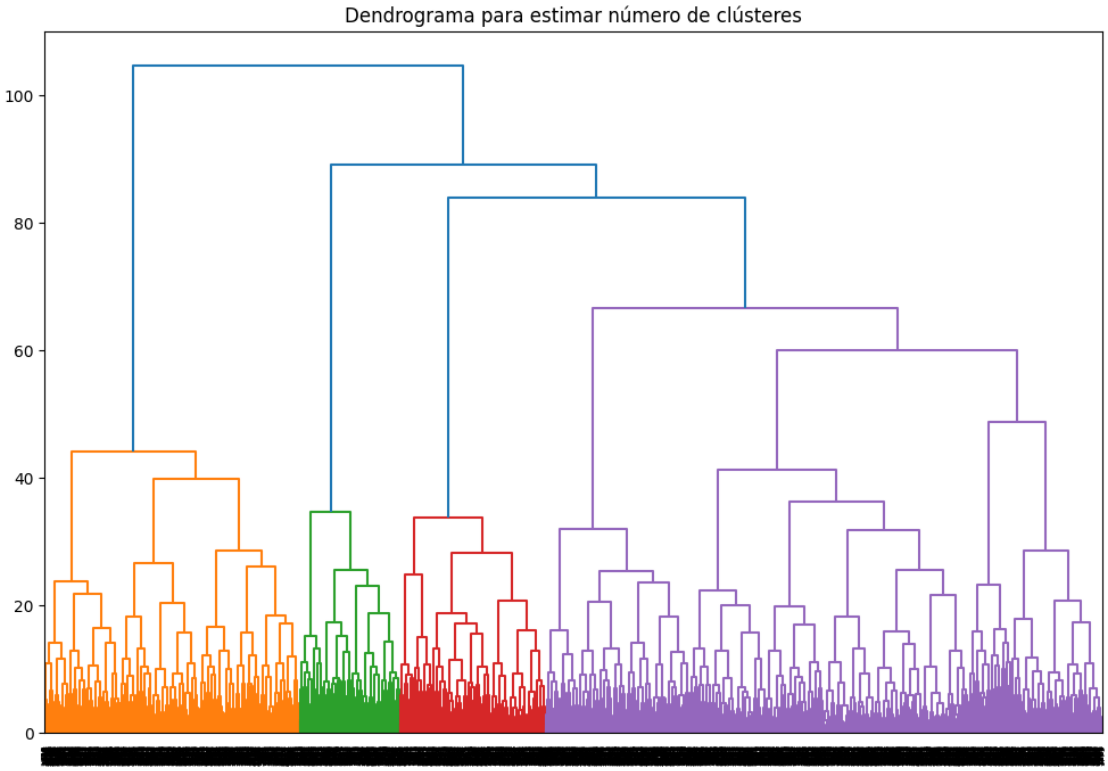
17 Paso 4: Creación de Clústeres

18 Estandarizar los datos antes de aplicar clustering

[94]:
```
X_scaled = scaler.fit_transform(X)
```

19 Dendrograma para estimar número de clústeres

[95]:
```
plt.figure(figsize=(12, 8))
linkage_matrix = linkage(X_scaled, method='ward')
dendrogram(linkage_matrix)
plt.title("Dendrograma para estimar número de clústeres")
plt.show()
```



20 K-Means con 5 clústeres

[96]:
```
kmeans = KMeans(n_clusters=5, random_state=42)
df['Cluster'] = kmeans.fit_predict(X_scaled)
```

21 Análisis de clústeres

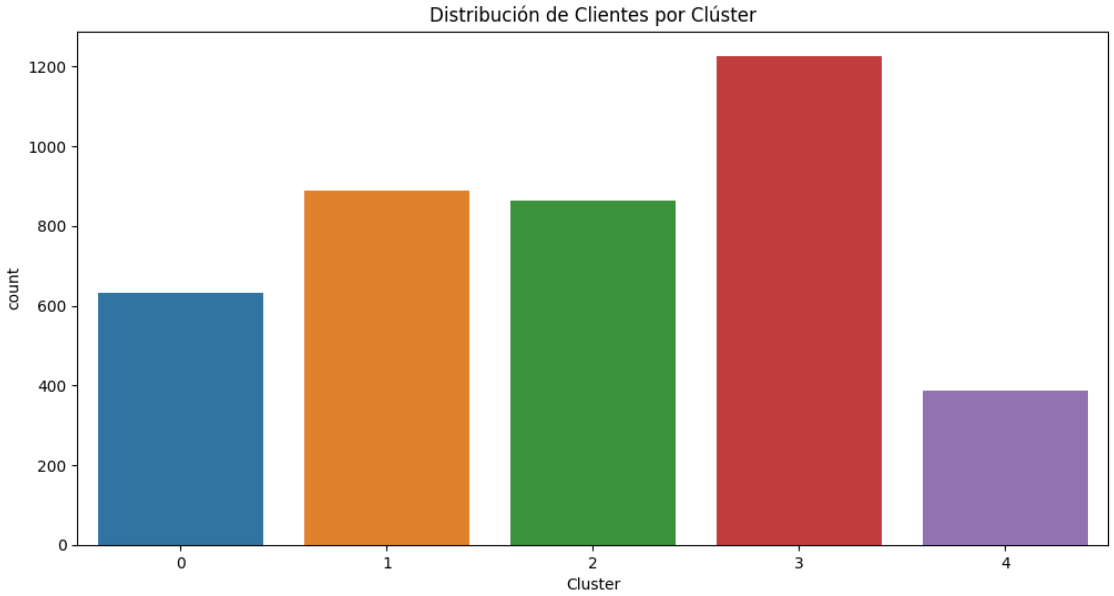
[97]:
```
display(df.groupby("Cluster").mean())
```

```
           gender  Near_Location   Partner  Promo_friends  Phone  \
Cluster
0        0.502370       0.949447  0.829384       0.998420    1.0
1        0.554556       0.849269  0.263217       0.052868    1.0
2        0.499422       0.937572  0.737572       0.478613    1.0
3        0.485738       0.718011  0.299919       0.018745    1.0
4        0.523316       0.862694  0.471503       0.305699    0.0

         Contract_period  Group_visits        Age  \
Cluster
0               3.097946      0.448657  29.104265
1               2.606299      0.436445  30.008999
```

```
2              11.854335      0.546821  29.905202
3               1.914425      0.276284  28.083945
4               4.777202      0.427461  29.297927

         Avg_additional_charges_total  Month_to_end_contract  Lifetime  \
Cluster
0                          141.774331              2.887836  3.772512
1                          159.774265              2.419573  4.780652
2                          163.509804             10.812717  4.677457
3                          129.496414              1.821516  2.195599
4                          144.208179              4.466321  3.940415

         Avg_class_frequency_total  Avg_class_frequency_current_month  \
Cluster
0                         1.770536                           1.669581
1                         2.745505                           2.733173
2                         2.006405                           1.998179
3                         1.225192                           0.967967
4                         1.854211                           1.723967

            Churn
Cluster
0        0.246445
1        0.089989
2        0.021965
3        0.572942
4        0.266839
```

22 Visualización de características por clúster

[104]:
```
plt.figure(figsize=(12, 6))
sns.countplot(x="Cluster", data=df)
plt.title("Distribución de Clientes por Clúster")
plt.show()
```



Comentario del revisor (1ra Iteracion)

Bien hecho! Se nota claramente la segmentación entre los diferentes grupos ya que tienen un churn rate diferente

23 Paso 5: Conclusiones y Recomendaciones

23.1 Segmentación de clientes:

Se identificaron clústeres con diferentes patrones de comportamiento y tasas de cancelación. Los clientes con menor tiempo de membresía y menor frecuencia de visitas son más propensos a abandonar el gimnasio.

23.2 Estrategias de retención

Implementar programas de fidelización dirigidos a los clústeres con alta tasa de cancelación. Ofrecer descuentos o beneficios a los clientes en riesgo para incentivar su permanencia.

23.3 Personalización de la comunicación

Contactar proactivamente a clientes con menor asistencia mediante recordatorios y ofertas personalizadas. Usar campañas de marketing específicas para cada clúster, resaltando los beneficios más atractivos según sus preferencias.

23.4 Optimización de servicios

Mejorar la experiencia de los clientes mediante entrenadores personales o actividades grupales exclusivas. Incentivar a los clientes más fieles con programas de recompensas y promociones especiales.

23.5 Conclusion Final

Estas estrategias permitirán reducir la tasa de cancelación y mejorar la retención de clientes en Model Fitness

Comentario general (1ra Iteracion)

Muy buen trabajo, manejaste muy bien los datos previo al entrenamiento de tus modelos y analizaste muy bien los resultados.

Te felicito por las conclusiones planteadas luego de ver los resultados de la segmentación utilizando los cluster, un buen análisis siempre debe venir acompañado de recomendaciones que ayuden a la toma de decisiones que en este caso haz planteado sugerencias muy buenas en cuanto a los diferentes tipos de usuarios obtenidos.

Saludos!