UVM Bootcamp Data Science

Proyecto Final

INFORME DE RESULTADOS DE UN ANÁLISIS PREDICTIVO DE ABANDONO EN TELEMARKETING

ALUMNO: GERARDO RODRÍGUEZ

INFORME DE RESULTADOS DE UN ANÁLISIS PREDICTIVO DE ABANDONO EN TELEMARKETING

Contenido

i. INTRODUCCIÓN

El presente Análisis busca identificar la mejor forma de predecir que personas podrían tener riesgo de abandonar su servicio de telefonía, basados en resultados históricos de una empresa de telefonía celular.

Por medio de Ciencia de datos se desarrolló un conjunto de modelos para identificar las causas del abandono de los clientes y proponer soluciones de negocio de una empresa telefónica.

Los datos históricos del cliente han sido extraídos y guardados en el archivo CustomerChurn.csv y con estos se realizaron las siguientes actividades:

- Análisis exploratorio de datos (entender las tendencias, patrones y anomalías)
- Desarrollo de tres modelos predictivos diferentes
- Selección del modelo más eficaz y justificación de la elección
- Elaboración de un reporte detallado de los hallazgos y recomendaciones

La información histórica contiene las siguientes variables:

- Call Failures: número de fallas en llamadas
- Complains: variable binaria que indica se hay historial de quejas (0: No hay quejas, 1: Quejas)
- Subscription Length: total de meses de suscripción
- Charge Amount: atributo ordinal (0: mínimo monto, 9: máximo monto)
- Seconds of Use: total de segundos de llamadas
- Frequency of use: número total de llamadas
- Frequency of SMS: número total de mensajes de texto
- Distinct Called Numbers: total de números distintos a los que ha llamado
- Age Group: atributo ordinal (1: edad más joven, 5: mayor edad)
- Tariff Plan: variable binaria (1: Pago por uso, 2: Contractual)
- Status: variable binaria (1: activo, 2: inactivo)
- Churn: variable binaria (1: abandono, 0: no abandono) – Etiqueta de clase
- Customer Value: Valor estimado del cliente

ii. ANÁLISIS EXPLORATORIO (EDA)

a) PREPARACIÓN DEL PROYECTO

Se utilizarán las siguientes librerías:

- pandas
- numpy
- matplotlib.pyplot
- matplotlib
- seaborn
- plotly.express
- warnings

Se transforman las cabeceras para un mayor entendimiento de los datos:

- Call Failures: 'llamadasFallidas'
- Complains: 'quejas'
- Length: 'mesesSubscripcion'
- Subscription Charge Amount: 'nivelMonto'
- Seconds of Use: 'segundosUtilizados'
- Frequency of use: 'frecuenciaDeUso'
- Frequency of SMS: 'usoSms'
- Distinct Called Numbers: 'numerosMarcadosUnicos'
- Age Group: 'grupoDeEdad'
- Tariff Plan: 'planTarifario'
- Status: 'estatus'
- Churn: 'abandono'
- Age: 'edad'
- Customer Value: 'valorCliente'

b) ANÁLISIS EXPLORATORIO DE DATOS I

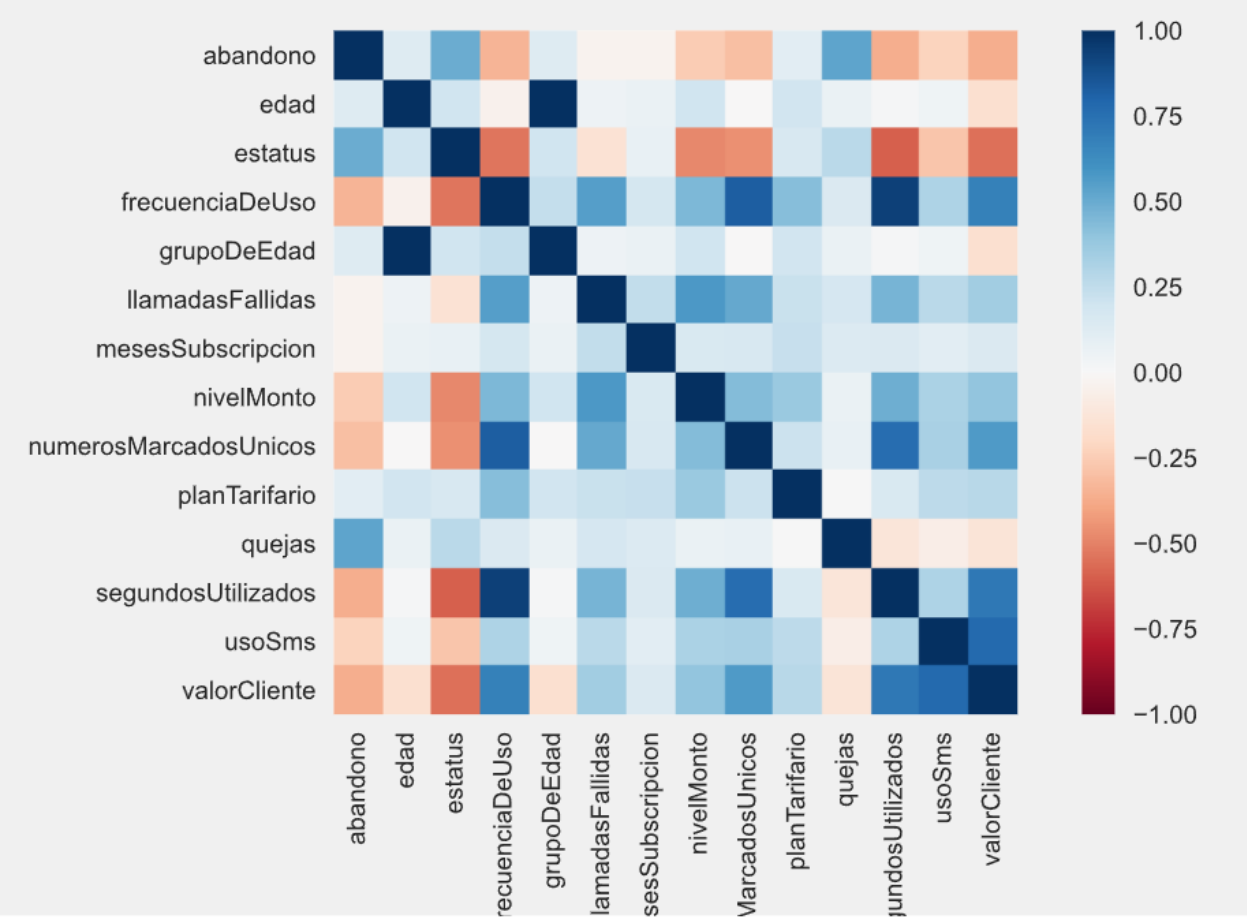
Se realizará un análisis exploratorio de datos inicial para entender la estructura general de la data, posteriormente se realizará un segundo bloque de análisis para revisar la información con datos etiquetados.

HALLAZGOS REPORTE YDATA PROFILING

Se identifican las siguientes correlaciones en la data:

- abandono está altamente correlacionado con quejas - ALTA CORRELACIÓN
- edad está altamente correlacionado con grupoDeEdad - ALTA CORRELACIÓN
- estatus está altamente correlacionado con frecuenciaDeUso - ALTA CORRELACIÓN
- frecuenciaDeUso está altamente correlacionado con estatus - ALTA CORRELACIÓN
- grupoDeEdad está altamente correlacionado con edad - ALTA CORRELACIÓN
- llamadasFallidas está altamente correlacionado con frecuenciaDeUso - ALTA CORRELACIÓN
- nivelMonto está altamente correlacionado con llamadasFallidas - ALTA CORRELACIÓN
- numerosMarcadosUnicos está altamente correlacionado con frecuenciaDeUso - ALTA CORRELACIÓN
- quejas está altamente correlacionado con abandono - ALTA CORRELACIÓN
- segundosUtilizados está altamente correlacionado con estatus - ALTA CORRELACIÓN
- usoSms está altamente correlacionado con valorCliente - ALTA CORRELACIÓN
- valorCliente está altamente correlacionado con estatus - ALTA CORRELACIÓN

Esto da sospechas de que una parte del abandono puede venir de las quejas en el servicio, sin embargo más adelante identificaremos que tal vez no es tan relevante.



Patrones de datos:

- No se identifican valores nulos.
- Se identifican 7 registros con rangos de 4 a 6 duplicados para varios usuarios, pero al ser tantos datos podemos considerar posible en que se den casualidades.

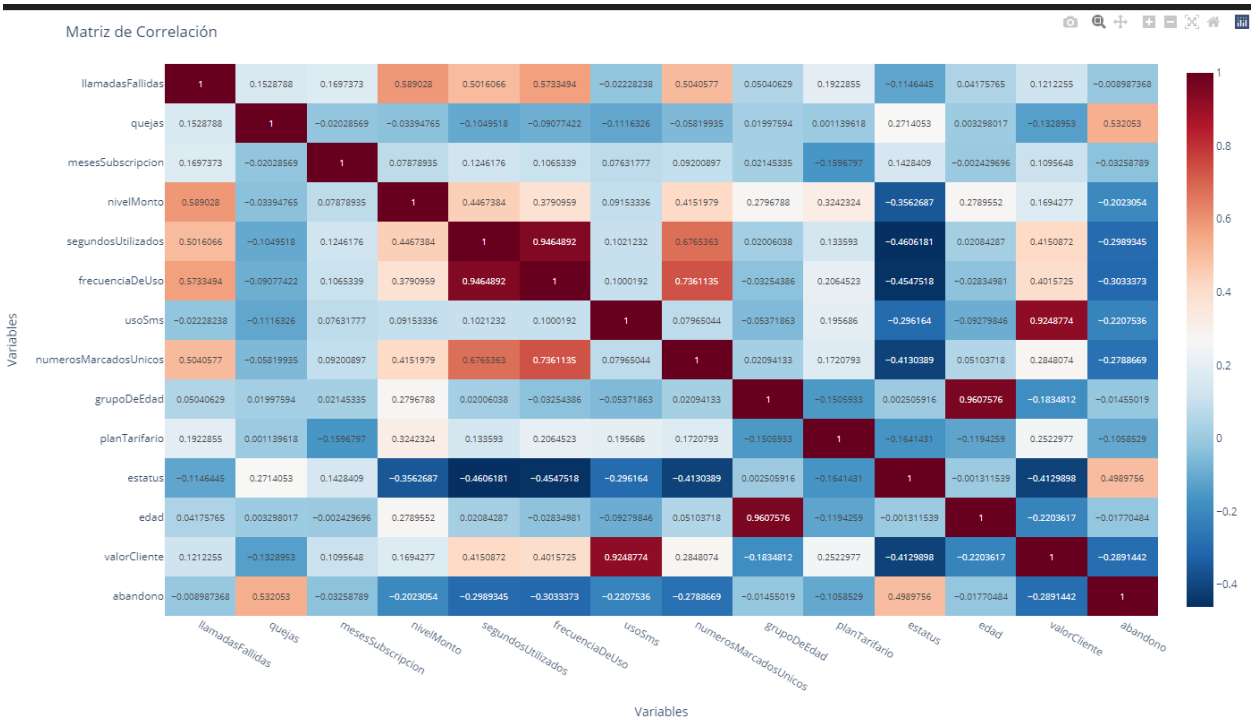
HALLAZGOS CORRELACIÓN PLOTLY EXPRESS

Igual que con YDATA se identifican las siguientes correlaciones de ABANDONO en la data:

- Quejas 0.53 y Estatus 0.49

Otras correlaciones interesantes son:

- El uso de SMS con el Valor del Cliente supera otros valores como segundos utilizados, nivel del monto o Frecuencia de uso.
- Algunas son obvias como segundos utilizados, nivel del monto o Frecuencia de uso con los Números marcados únicos.



c) PRE-PROCESAMIENTO Y LIMPIZA DE DATOS

Se identifica el formato de origen de los datos.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3150 entries, 0 to 3149
Data columns (total 14 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   llamadasFallidas       3150 non-null   int64
 1   quejas                 3150 non-null   int64
 2   mesesSubscripcion      3150 non-null   int64
 3   nivelMonto             3150 non-null   int64
 4   segundosUtilizados     3150 non-null   int64
 5   frecuenciaDeUso        3150 non-null   int64
 6   usoSms                 3150 non-null   int64
 7   numerosMarcadosUnicos  3150 non-null   int64
 8   grupoDeEdad            3150 non-null   int64
 9   planTarifario          3150 non-null   int64
 10  estatus                3150 non-null   int64
 11  edad                   3150 non-null   int64
 12  valorCliente           3150 non-null   float64
 13  abandono               3150 non-null   int64
dtypes: float64(1), int64(13)
memory usage: 344.7 KB
```

Se agregan dos variables para comprender mejor la información.

```
# b. Agregar columna Minutos utilizados
df['minutosUtilizados'] = df['segundosUtilizados'] / 60
```

✓ 0.0s

```
# c. Agregar columna Anos de Suscripcion
df['anosSubscripcion'] = df['mesesSubscripcion'] / 12
```

✓ 0.0s

Se cambian algunas variables a categoría para poderlas visualizar en otro tipo de gráficas.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3150 entries, 0 to 3149
Data columns (total 16 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   llamadasFallidas       3150 non-null   int64
 1   quejas                 3150 non-null   category
 2   mesesSubscripcion      3150 non-null   int64
 3   nivelMonto             3150 non-null   category
 4   segundosUtilizados     3150 non-null   int64
 5   frecuenciaDeUso        3150 non-null   int64
 6   usoSms                 3150 non-null   int64
 7   numerosMarcadosUnicos  3150 non-null   int64
 8   grupoDeEdad            3150 non-null   category
 9   planTarifario          3150 non-null   category
 10  estatus                3150 non-null   category
 11  edad                   3150 non-null   category
 12  valorCliente           3150 non-null   float64
 13  abandono               3150 non-null   category
 14  minutosUtilizados      3150 non-null   float64
 15  anosSubscripcion       3150 non-null   float64
dtypes: category(7), float64(3), int64(6)
memory usage: 244.4 KB
```

Se identifican las variables categóricas y sus valores numéricos

```
# f. Buscando valores unicos en columnas categoricas
for col in cat_cols:
    print(f"{col} tiene {df[col].unique()} valores\n")
```

✓ 0.0s

```
quejas tiene [0, 1]
Categories (2, int64): [0, 1] valores

nivelMonto tiene [0, 1, 2, 3, 8, ..., 9, 7, 5, 10, 6]
Length: 11
Categories (11, int64): [0, 1, 2, 3, ..., 7, 8, 9, 10] valores

grupoDeEdad tiene [3, 2, 1, 4, 5]
Categories (5, int64): [1, 2, 3, 4, 5] valores

planTarifario tiene [1, 2]
Categories (2, int64): [1, 2] valores

estatus tiene [1, 2]
Categories (2, int64): [1, 2] valores

edad tiene [30, 25, 15, 45, 55]
Categories (5, int64): [15, 25, 30, 45, 55] valores

abandono tiene [0, 1]
Categories (2, int64): [0, 1] valores
```

Se generan etiquetas para interpretar la data de los campos categóricos en las gráficas

```
quejas has ['No hay quejas', 'Quejas']
Categories (2, object): ['No hay quejas', 'Quejas'] values

nivelMonto has ['Nivel 0', 'Nivel 1', 'Nivel 2', 'Nivel 3', 'Nivel 8', ..., 'Nivel 9', 'Nivel 7', 'Nivel 5', 'Nivel 10', 'Nivel 6']
Length: 11
Categories (11, object): ['Nivel 0', 'Nivel 1', 'Nivel 2', 'Nivel 3', ..., 'Nivel 7', 'Nivel 8', 'Nivel 9', 'Nivel 10'] values

grupoDeEdad has ['De 31 a 44', 'De 25 a 30', 'De 15 a 24', 'De 45 a 54', 'De 55 a 60']
Categories (5, object): ['De 15 a 24', 'De 25 a 30', 'De 31 a 44', 'De 45 a 54', 'De 55 a 60'] values

planTarifario has ['Pago por uso', 'Contractual']
Categories (2, object): ['Pago por uso', 'Contractual'] values

estatus has ['Activo', 'No Activo']
Categories (2, object): ['Activo', 'No Activo'] values

edad has ['30', '25', '15', '45', '55']
Categories (5, object): ['15', '25', '30', '45', '55'] values

abandono has ['No Abandono', 'Abandono']
Categories (2, object): ['No Abandono', 'Abandono'] values
```

d) ANÁLISIS EXPLORATORIO DE DATOS II

Ya con los datos categorizados, procedemos a realizar un segundo análisis de datos.

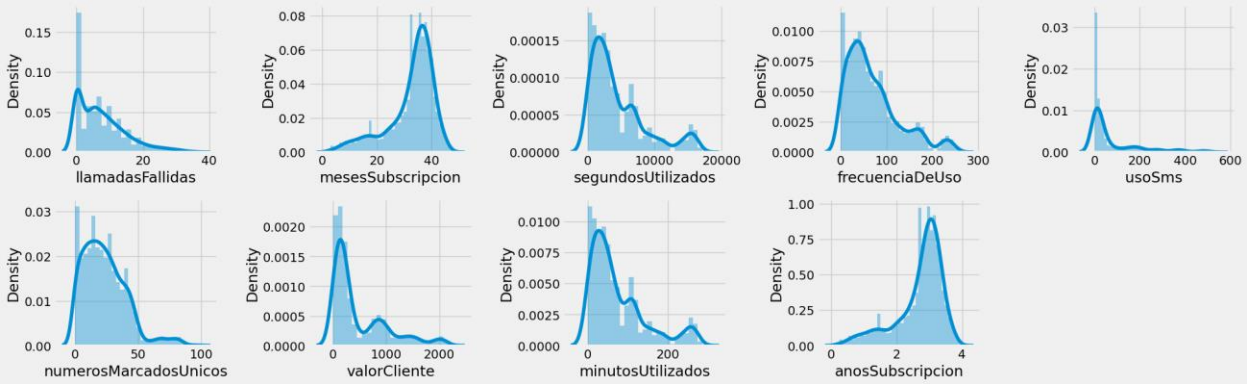
Haremos una revisión utilizando variables similares para buscar patrones.

En este caso dividieremos la información como se muestra en la siguiente tabla:

GRUPOS DE VARIABLES	VARIABLES
Demografico	numerosMarcadosUnicos
Demografico	grupoDeEdad
Demografico	edad
Fallas del Servicio	llamadasFallidas
Fallas del Servicio	quejas
Valor para la empresa	mesesSubscripcion
Valor para la empresa	nivelMonto
Valor para la empresa	valorCliente
Valor para la empresa	planTarifario
Uso del Servicio	estatus
Uso del Servicio	segundosUtilizados
Uso del Servicio	frecuenciaDeUso
Uso del Servicio	usoSms

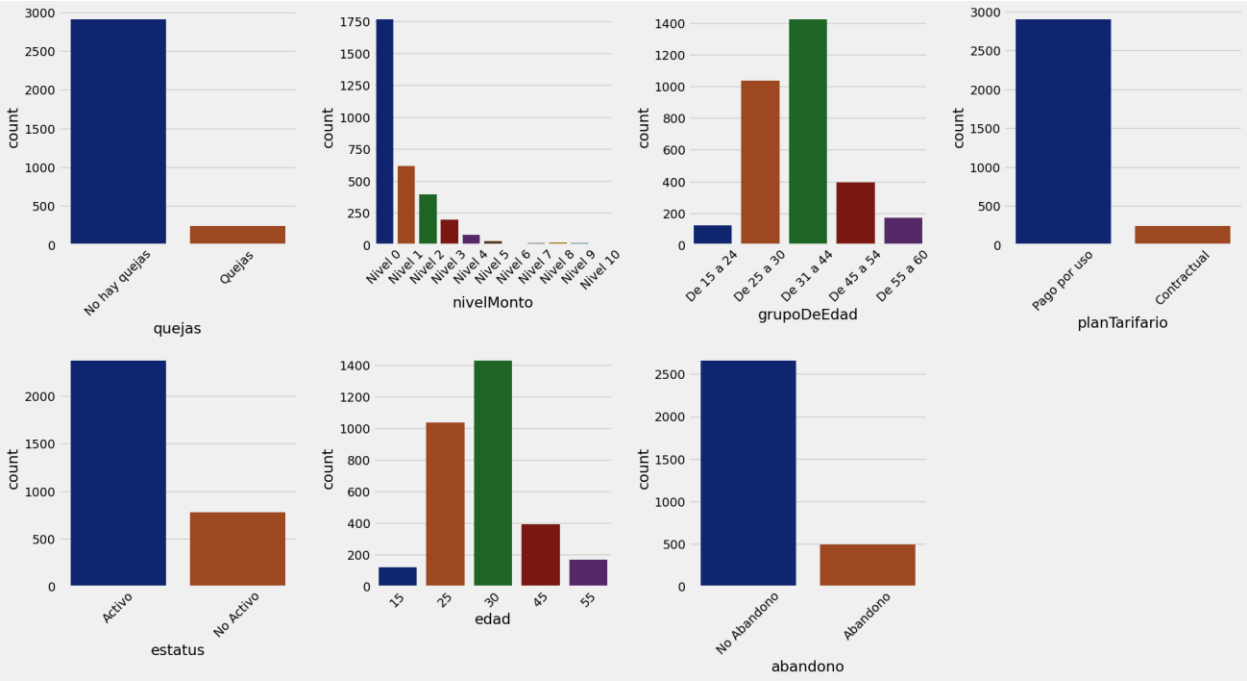
DATOS GENERALES

PLOTNUMBER – VARIABLES NUMÉRICAS



- Se identifica que la mayoría de los usuarios no pasa de 10 llamadas fallidas.
- En general la mayoría de los usuarios llegan al tercer año con su subscripción, pero perdemos a la mayoría en el año 1.
- Las curvas de uso (segundos, frecuencia, sms, números únicos) tienen una relación en la manera en que se distribuye la curva.

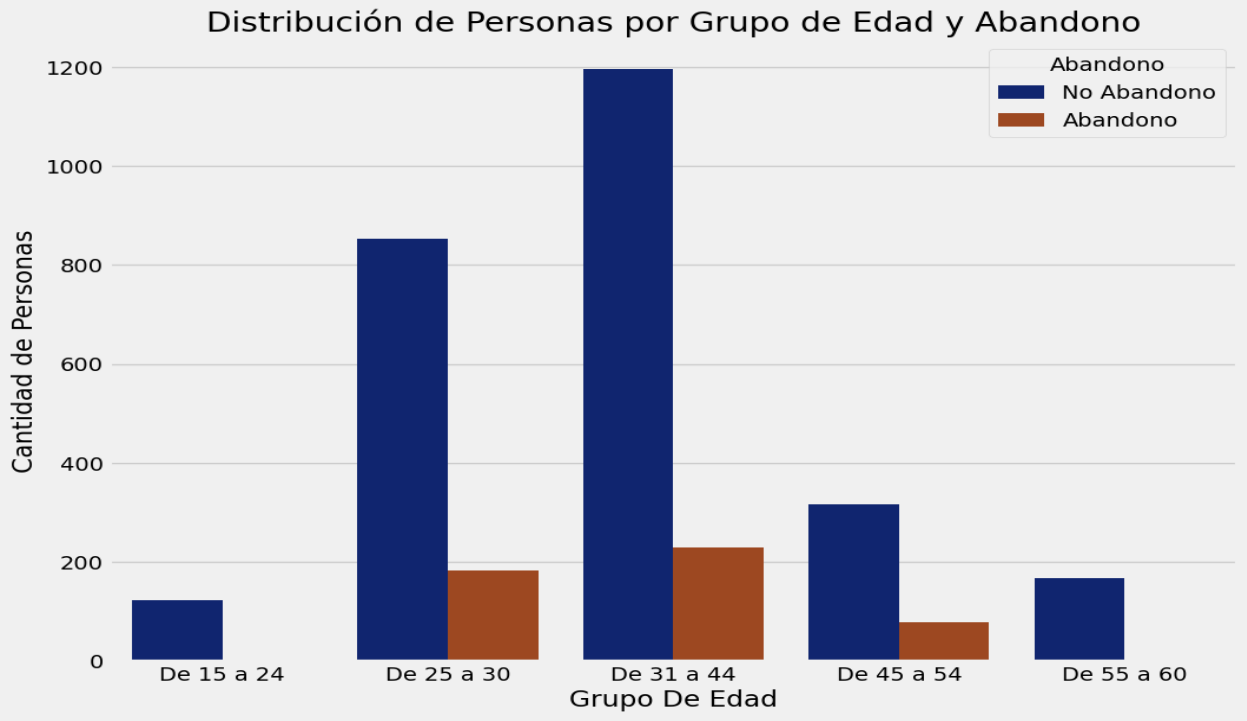
PLOTNUMBER – VARIABLES CATEGÓRICAS



- El volumen de quejas y usuarios por contrato fijo es inferior al de abandono, sin embargo, el abandono es más similar a los usuarios No Activos.
- La columna edad no tiene una distribución que manifieste datos reales, salde 15 a 25 y de 25 a 30 y así para adelante. El dato no parece relevante de esta forma.
- La relación de edad y grupo de edad es muy similar, podríamos prescindir de la columna edad, ya que para el modelo es más útil tener las categorías por grupos.
- La mayoría de nuestros usuarios se identifica de manera decreciente en los niveles de monto.

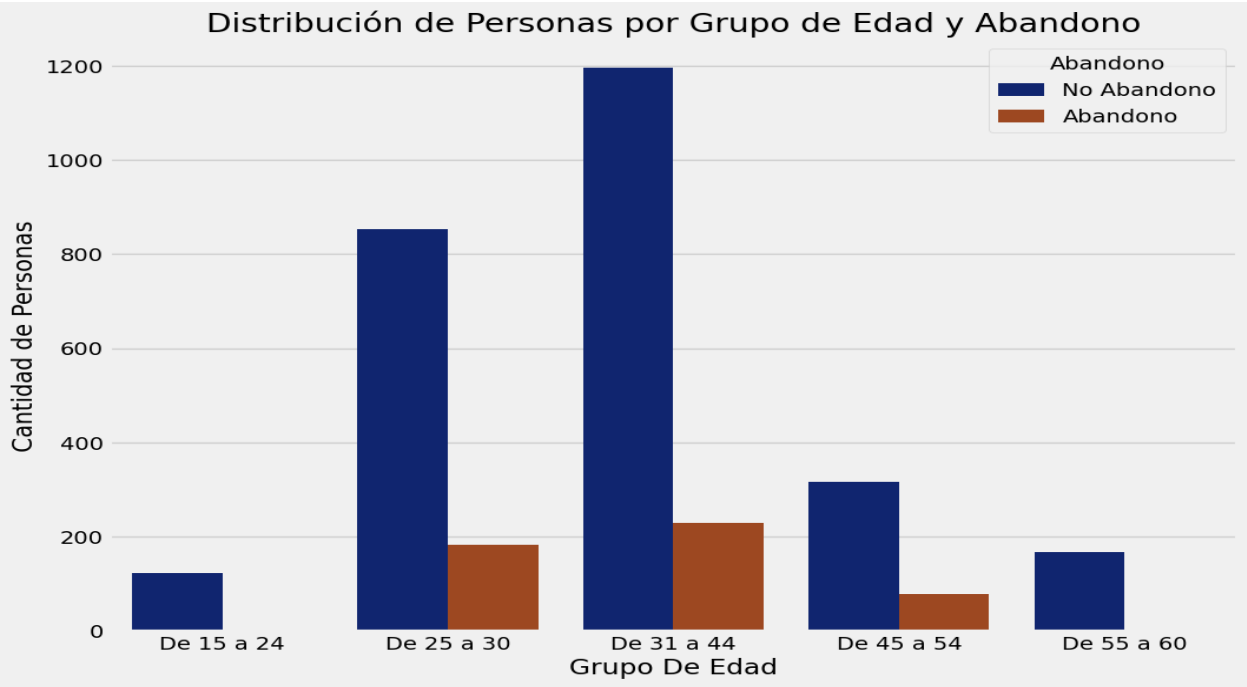
DATOS DEMOGRÁFICOS

ABANDONO VS. EDAD



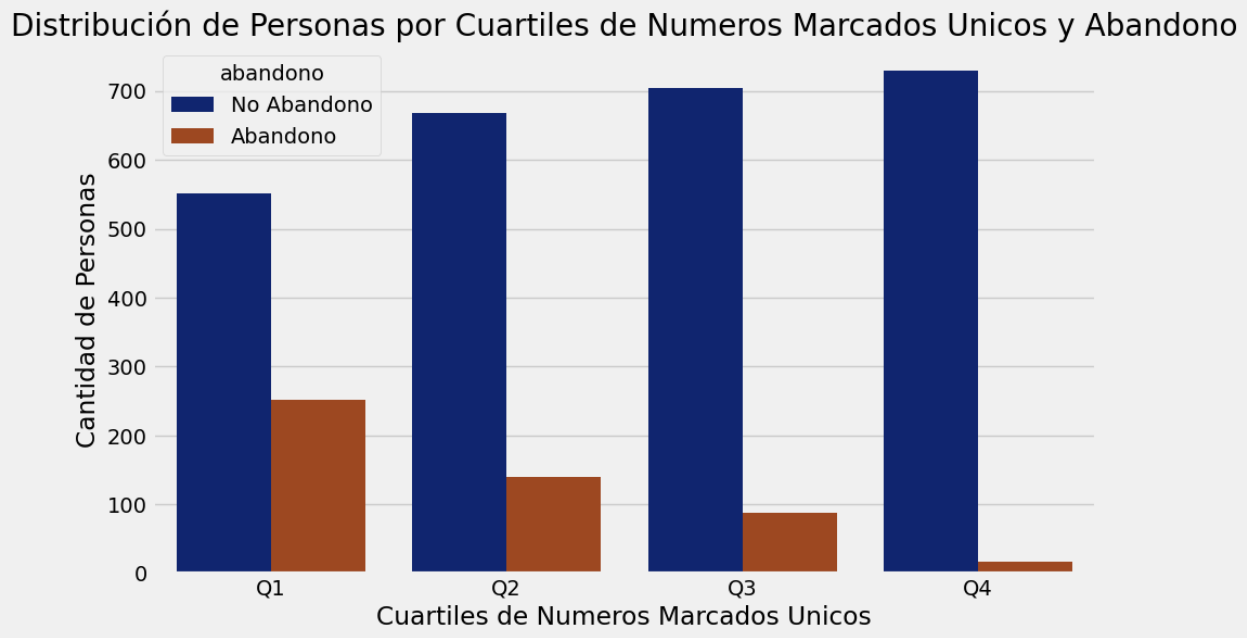
Es relevante que la mayoría de nuestros usuarios abandonan nuestro servicio en el rango de 25 a 54.

ABANDONO VS. GRUPO DE EDAD



Como hemos comentado esta gráfica se comporta prácticamente igual que la anterior, por lo que dejaremos de usar la columna edad.

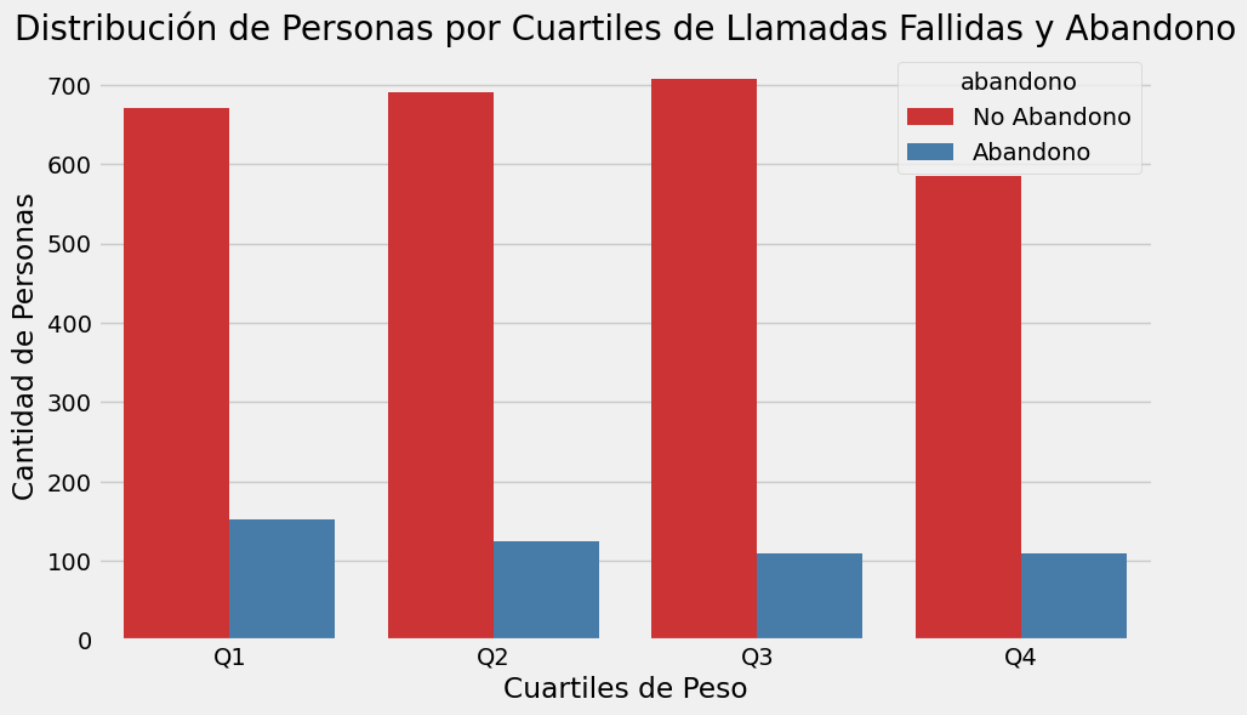
GRAFICO CUARTILES DE NUMEROS MARCADOS UNICOS VS. ABANDONO



No hay relación entre el abandono y las personas que tienen más contactos

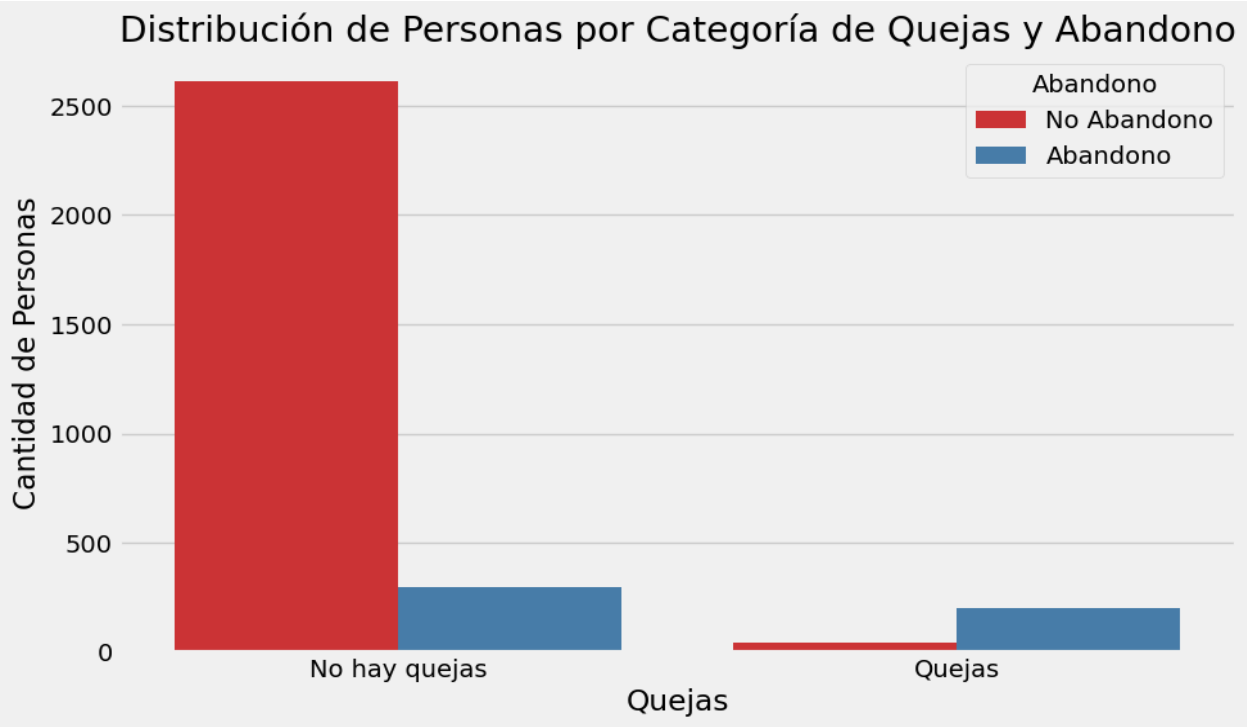
DATOS DE FALLAS DEL SERVICIO

GRAFICO CUARTILES DE LLAMADAS FALLIDAS VS. ABANDONO



Aunque existe una pequeña correlación entre abandono y llamadas fallidas, no parece contundente.

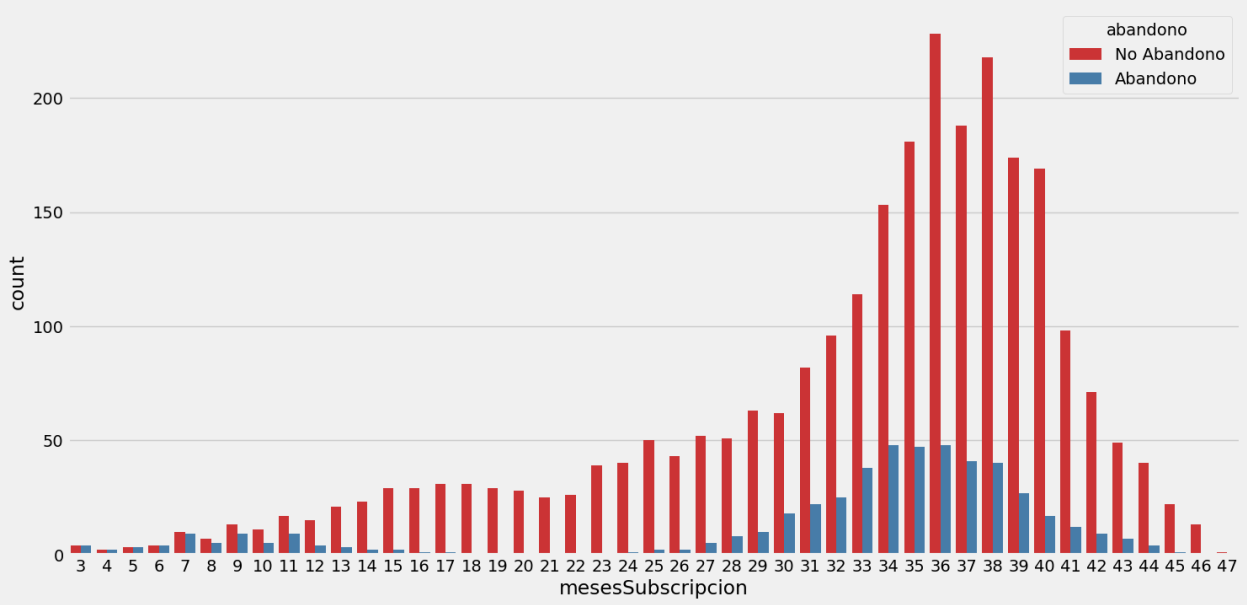
GRAFICO CUARTILES DE QUEJAS VS. ABANDONO



La relación entre quejas y abandono sigue sin ser radical.

DATOS VALOR DE LA EMPRESA

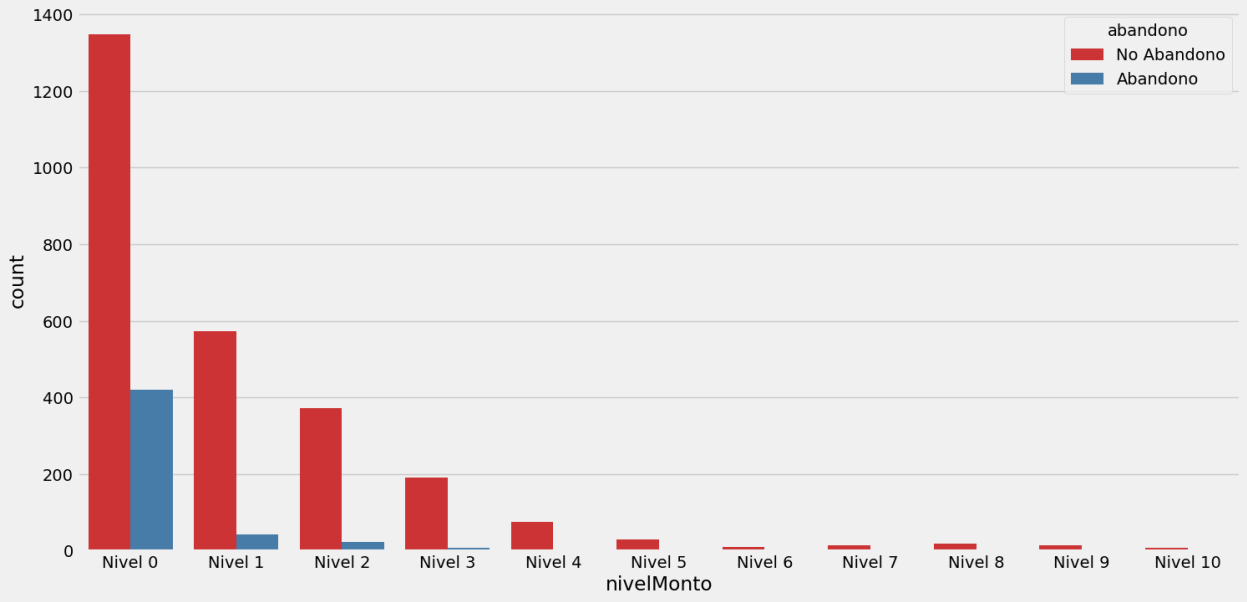
GRAFICO ABANDONO VS. MESES SUBSCRIPCION



Existe una correlación débil entre los usuarios nuevos (1 a 15 meses) respecto al abandono.

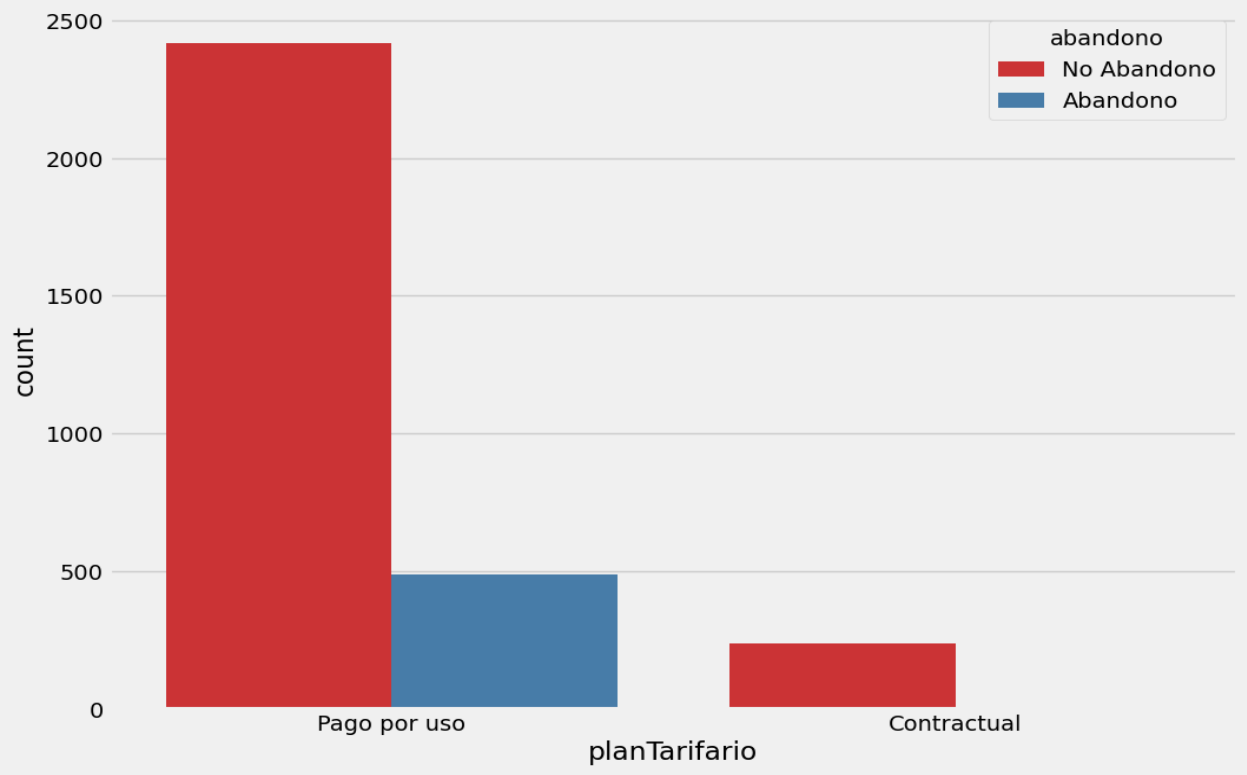
Y una correlación fuerte entre los usuarios más maduros (24 a 45 meses) respecto al abandono.

GRAFICO ABANDONO VS. NIVEL DE MONTO



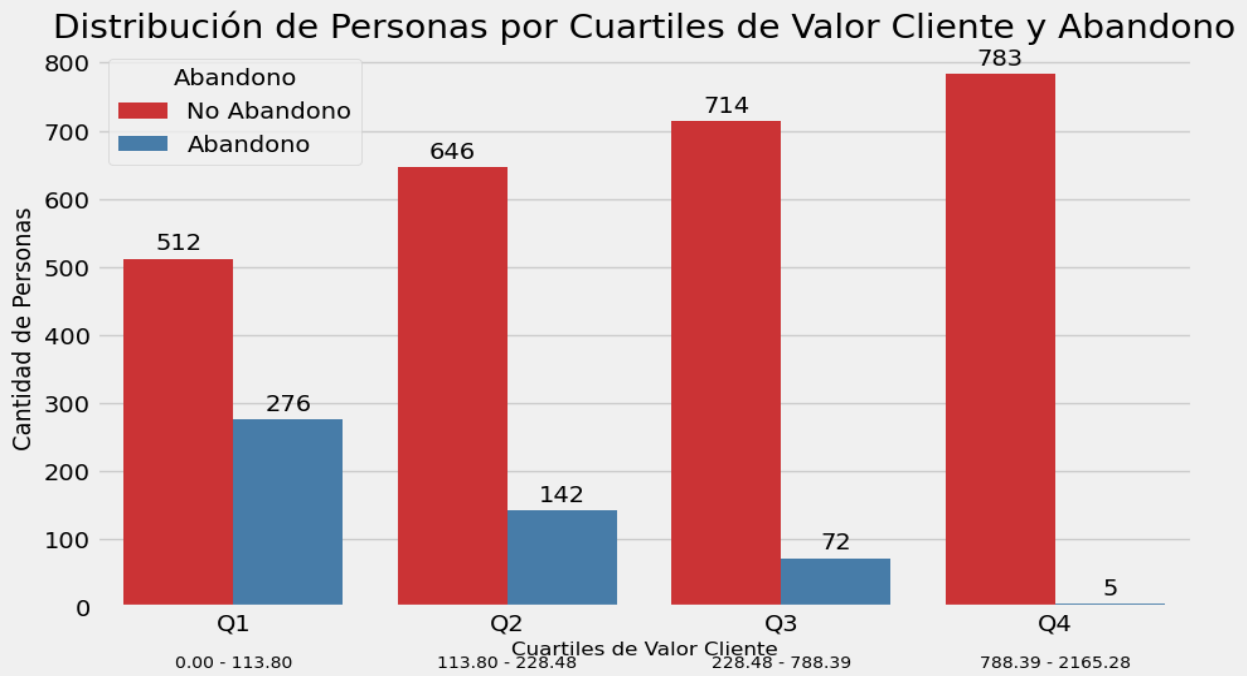
Correlación entre niveles de abandono y niveles de monto pagado por el cliente inferiores

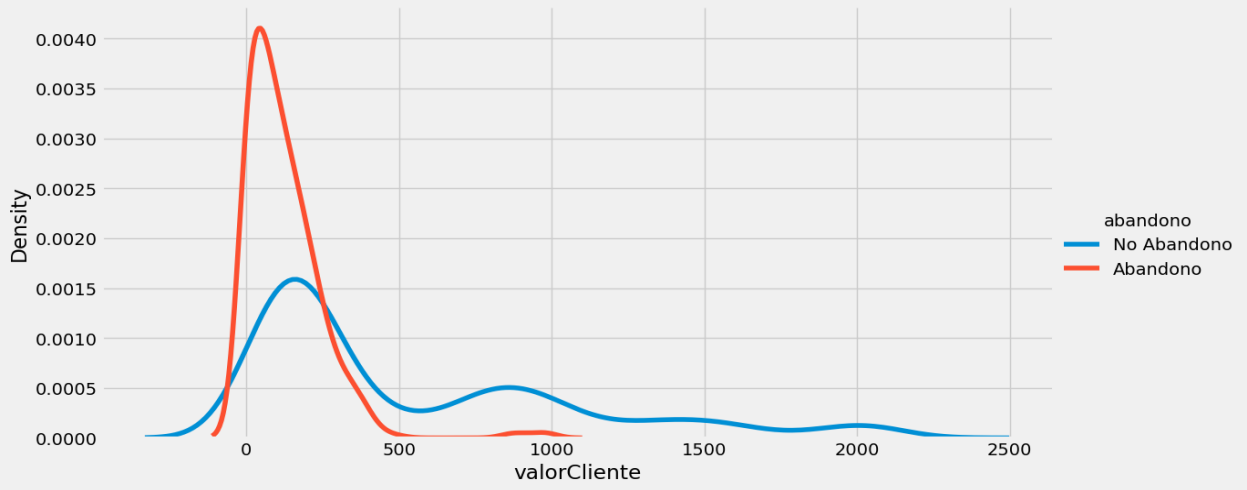
GRAFICO ABANDONO VS. PLAN TARIFARIO

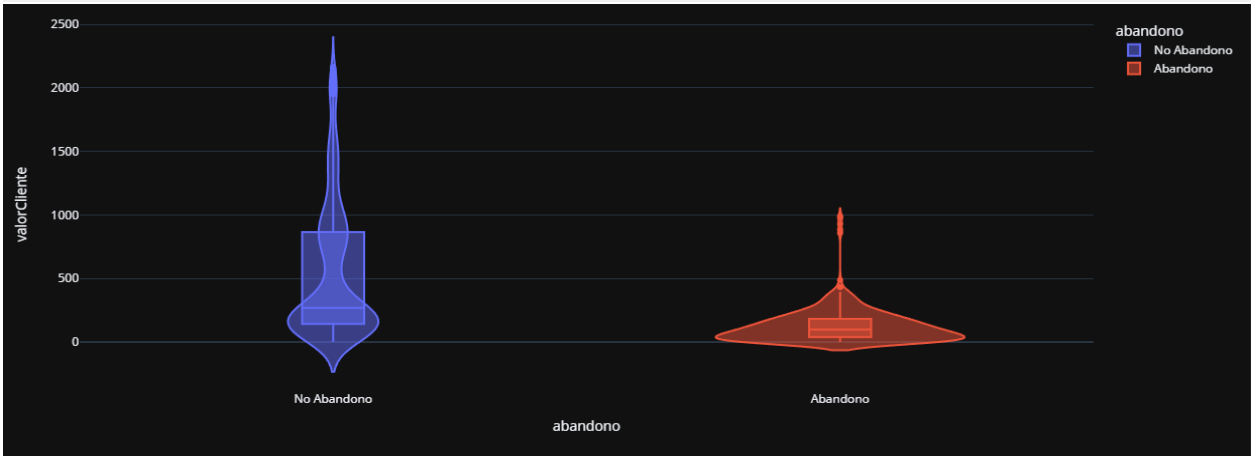


Correlación entre cliente de pago por uso y abandono.

GRAFICO CUARTILES DE VALOR DE CLIENTE VS. ABANDONO



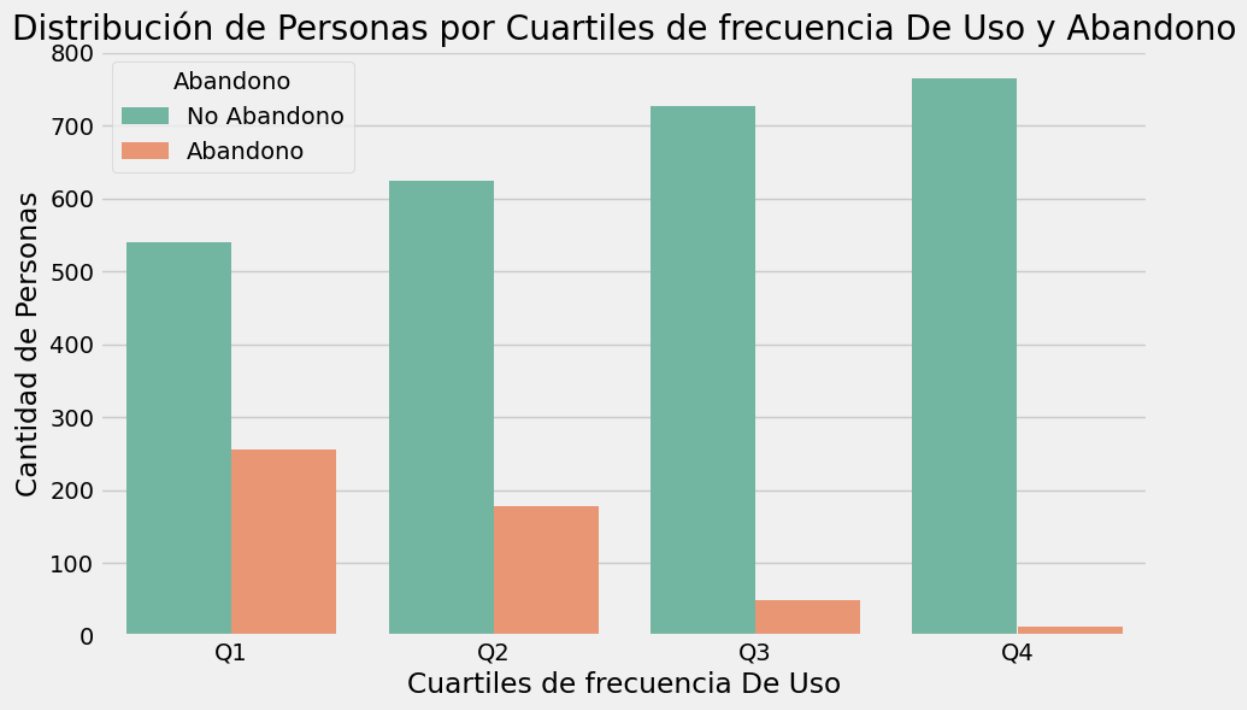




Correlación entre mayor abandono con los menores nivel de valor del cliente.

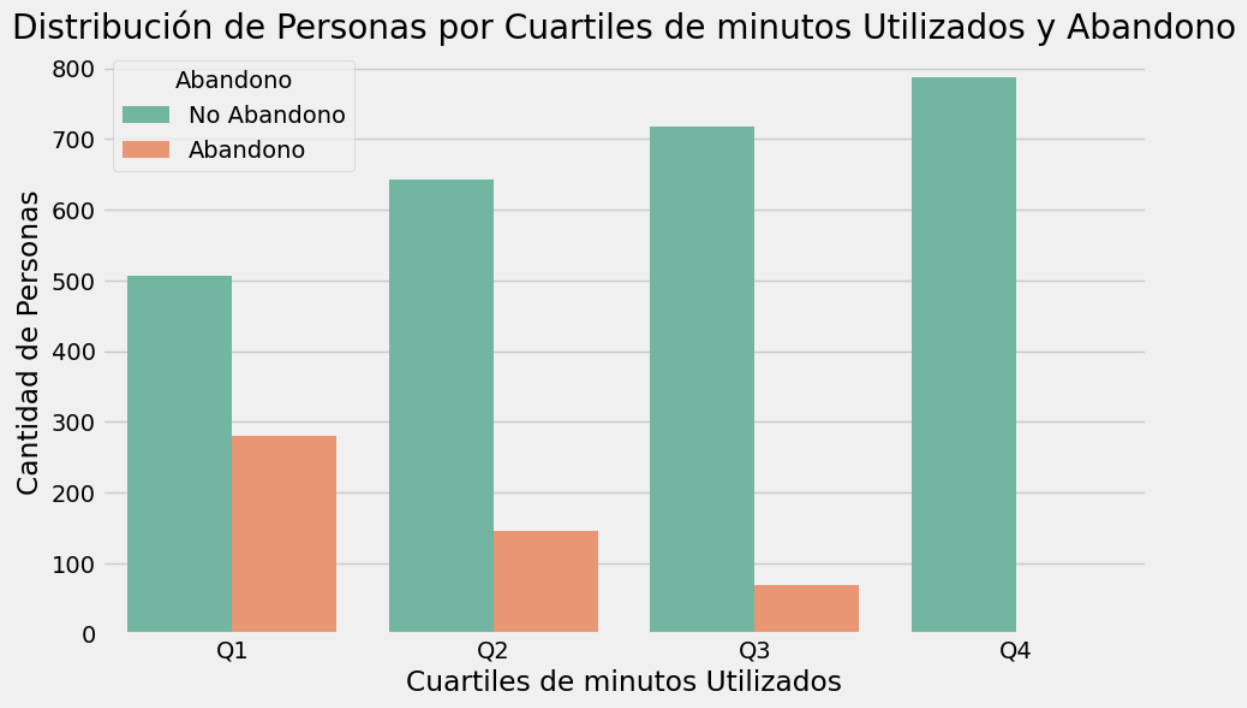
DATOS VALOR DE USO DEL SERVICIO

GRAFICO CUARTILES DE FRECUENCIA DE USO VS. ABANDONO



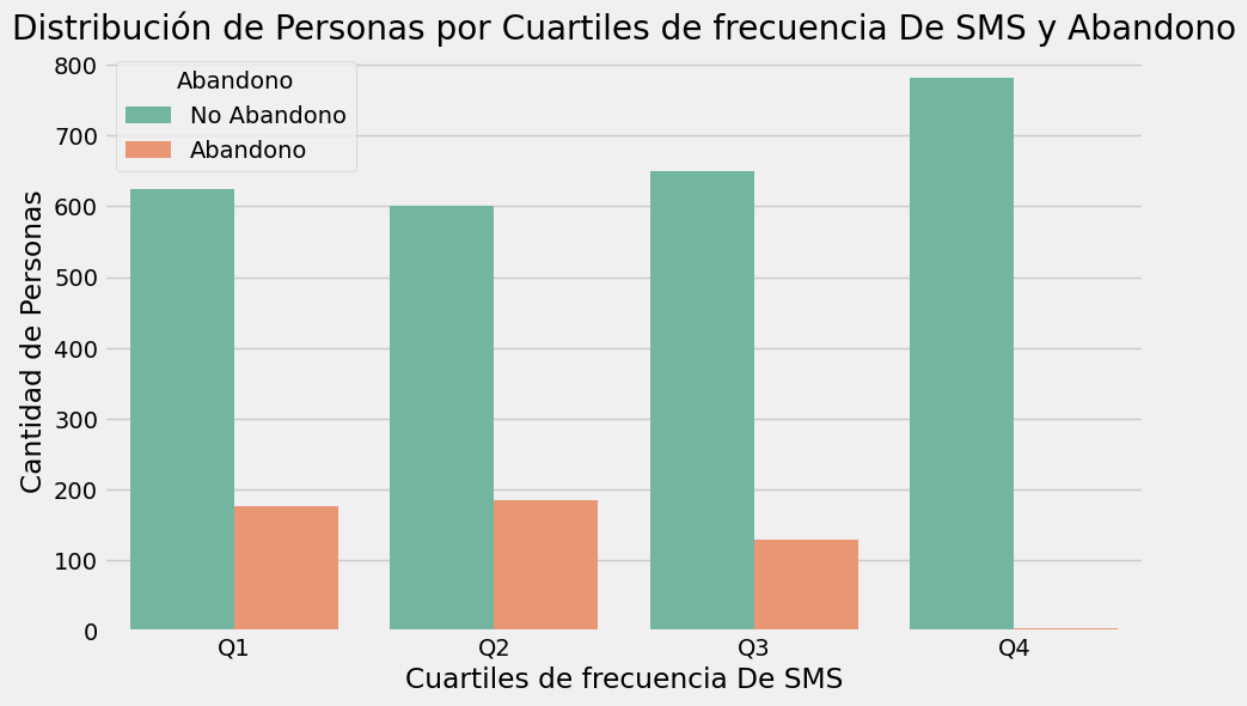
Relación Entre mayor Abandono y menor frecuencia de uso.

GRAFICO CUARTILES DE MINUTOS DE USO VS. ABANDONO



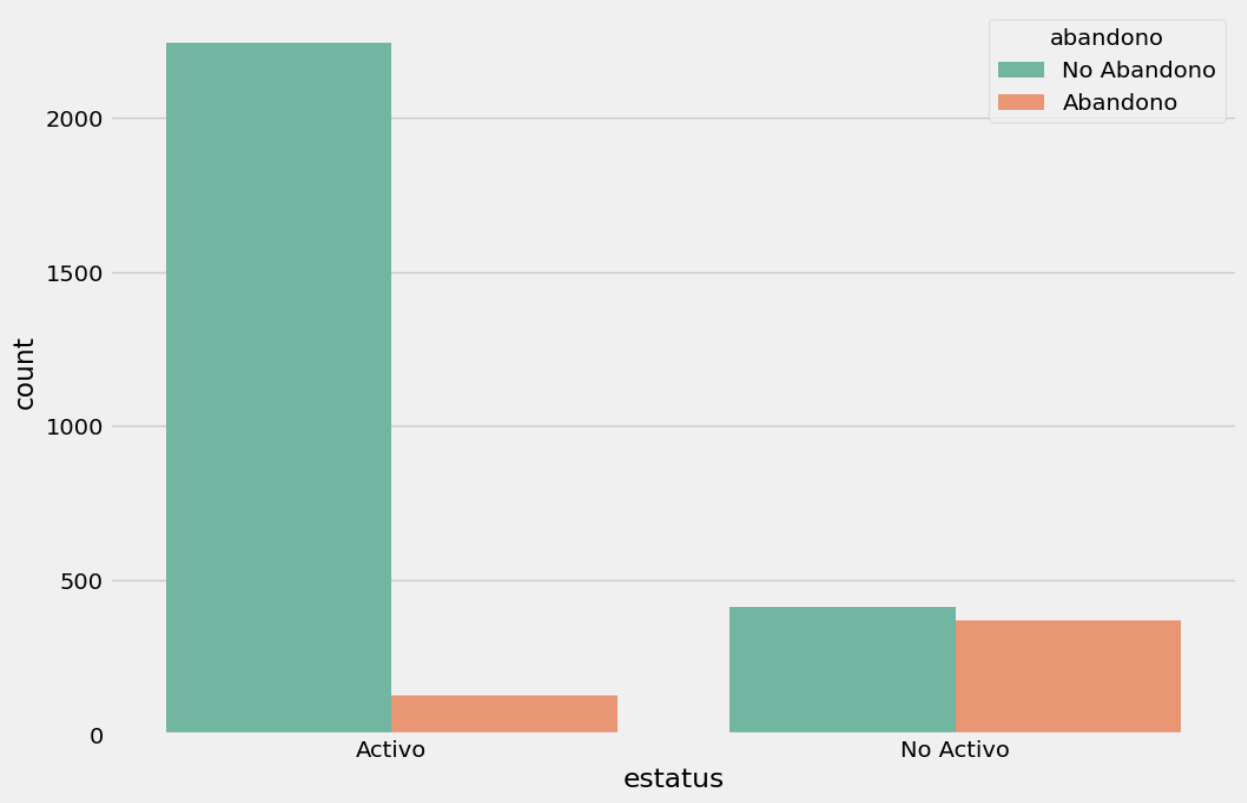
Relación entre mayor cantidad de minutos utilizados y abandono.

GRAFICO CUARTILES DE USO DE SMS VS. ABANDONO



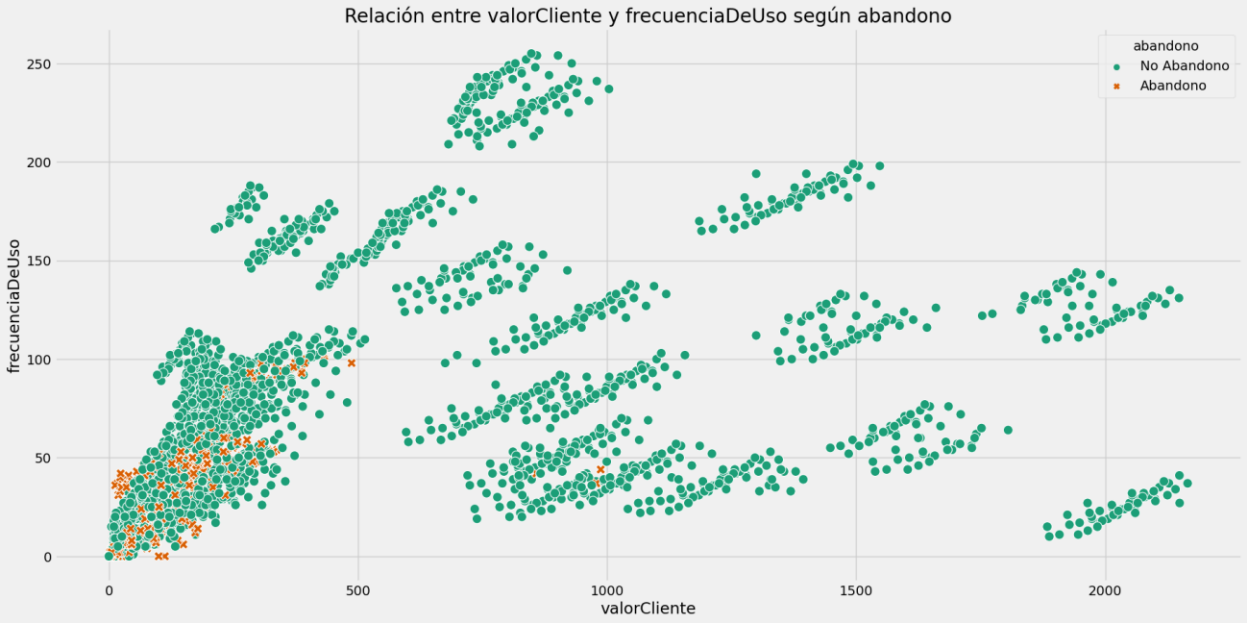
El uso de SMS es consistente entre los usuarios que usan menos mensajes SMS.

GRAFICO ABANDONO VS. ESTATUS



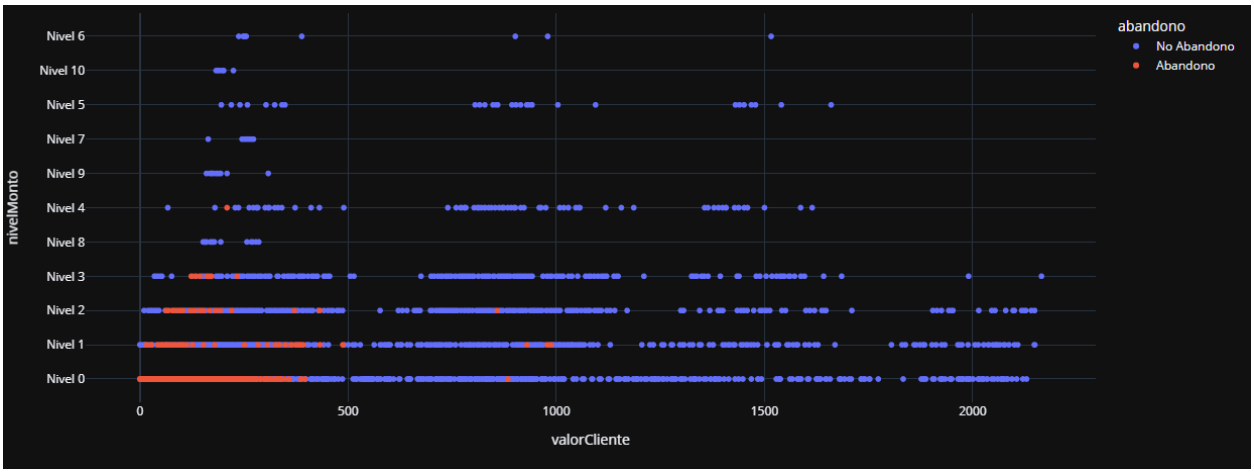
La correlación entre contratos inactivos es amplia en usuarios que abandonan.

GRAFICO ABANDONO VS. VALOR CLIENTE Y FRECUENCIA DE USO



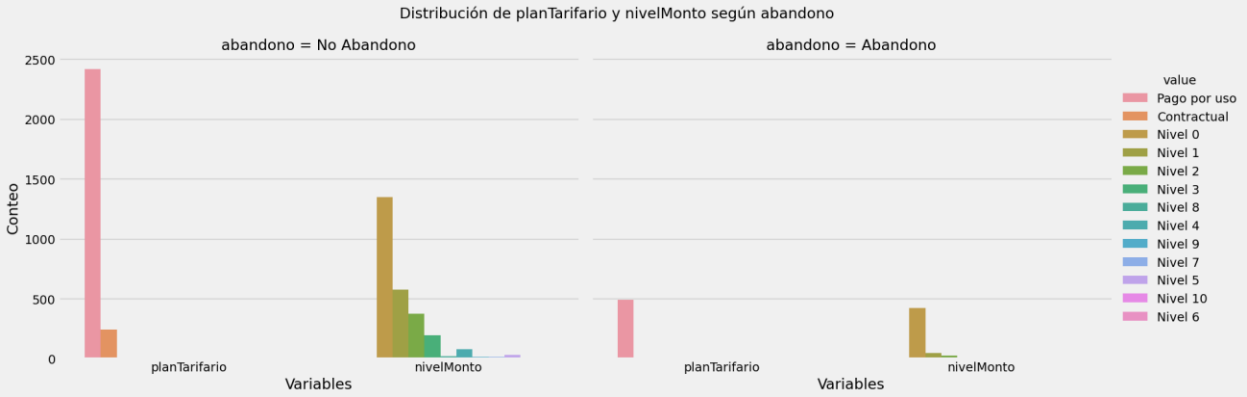
La frecuencia de uso tiene diferentes silos cuando se cruza con el valor del cliente.
Esto nos permite identificar una correlación entre los clientes de menor uso y con menor valor.

EJECUCION DE SCATTER EN VALORCLIENTE Y NIVELMONTO VS. ABANDONO



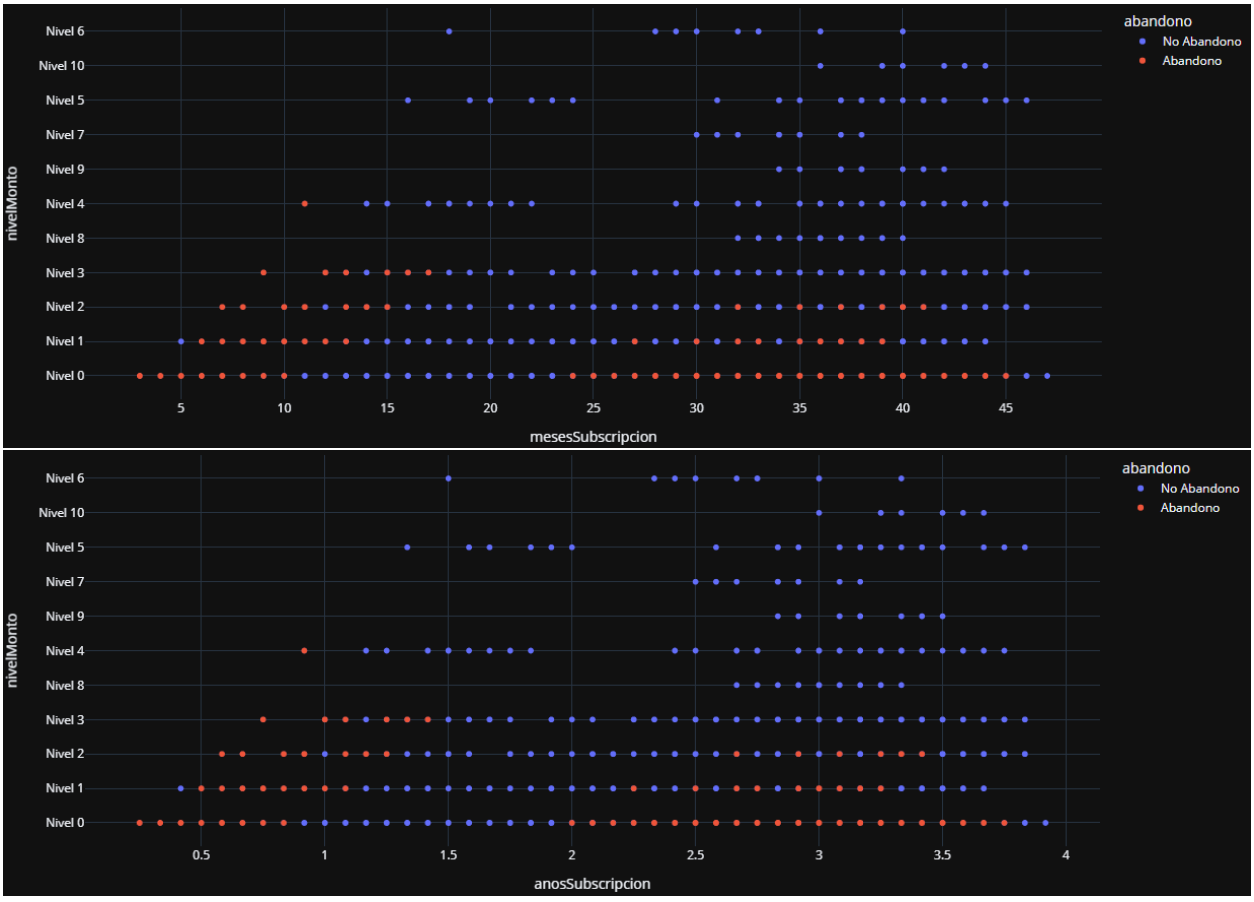
Al igual que con el gráfico anterior, el nivel de monto tiene diferentes silos cuando se cruza con el valor del cliente.
Esto nos permite identificar una correlación entre nivel de monto y valor de cliente con abandono.

GRAFICO PLAN TARIFARIO Y NIVEL DE MONTO VS ABANDONO



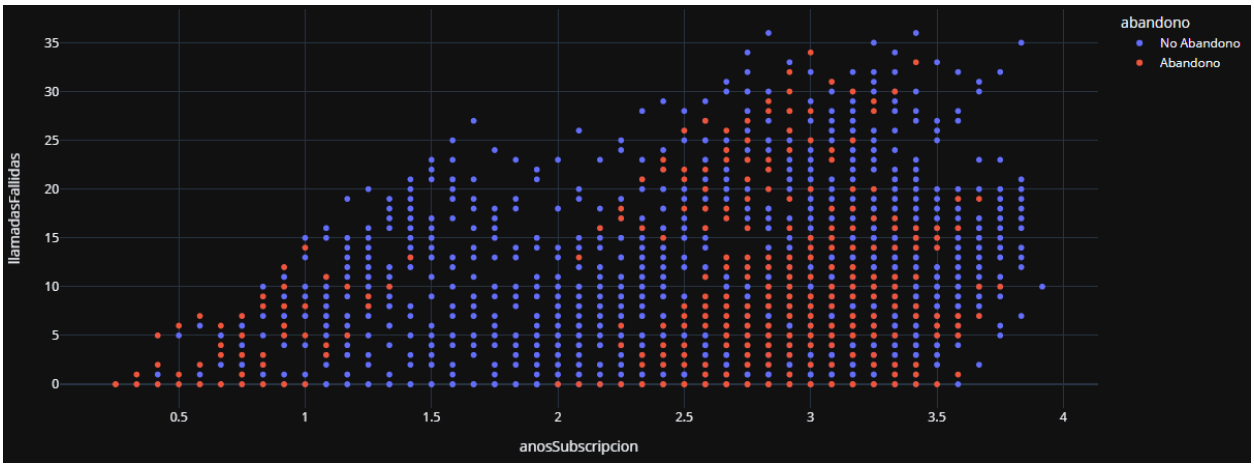
Identificamos que la mayoría de los abandonos vienen de usuarios en plan tarifario de pago por uso y los menores niveles de monto ingresado.

EJECUCION DE SCATTER EN TIEMPO DE SUBSCRIPCION Y NIVEL DE MONTO VS. ABANDONO



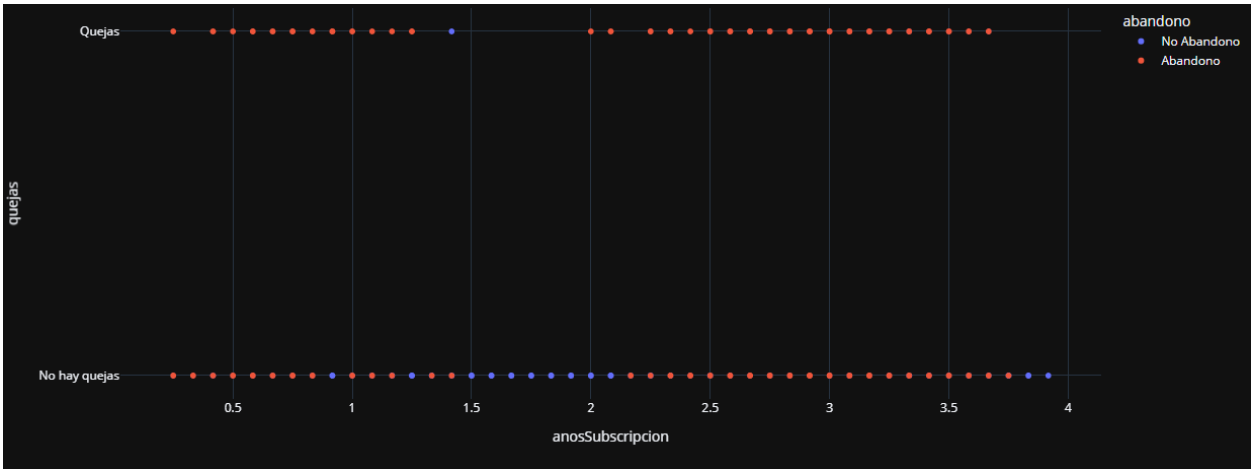
Los usuarios que abandonan son en su mayoría de nivel de ingresos 0 al 3, en rango de 0 a 1 año de antigüedad y de 2 a 3.5 años.

EJECUCION DE SCATTER EN TIEMPO DE SUBSCRIPCION Y LLAMADAS FALLIDAS VS. ABANDONO



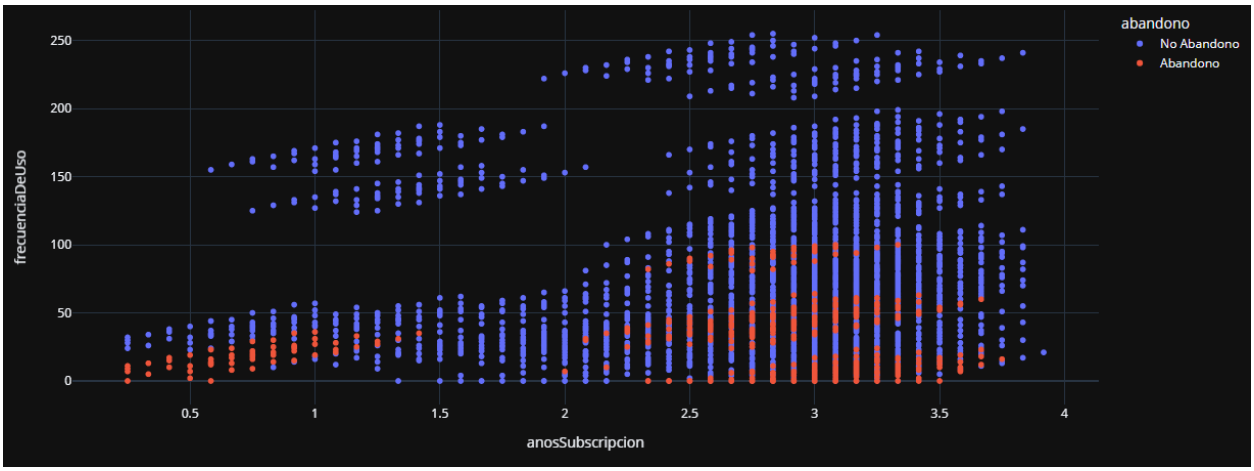
La correlación entre llamadas fallidas pinta en el mismo patron del gráfico anterior, lo que permite teorizar que los clientes abandonan en los niveles 1 a 3 de Monto por el número de fallas.

EJECUCION DE SCATTER EN TIEMPO DE SUBSCRIPCION Y QUEJAS VS. ABANDONO



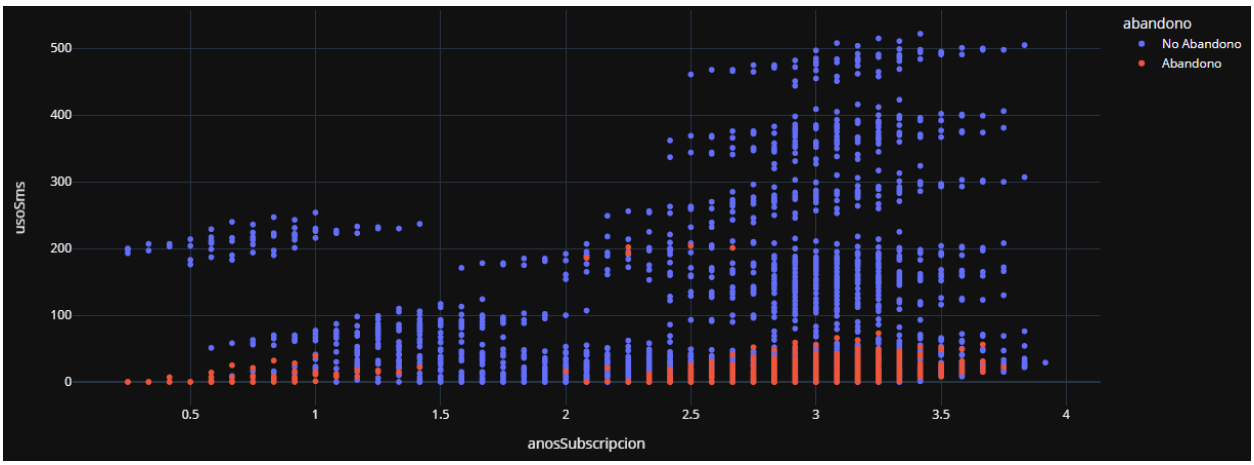
La cantidad de quejas es igual entre gente que abandona y gente que no abandona.

EJECUCION DE SCATTER EN TIEMPO DE SUBSCRIPCION Y FRECUENCIA DE USO VS. ABANDONO



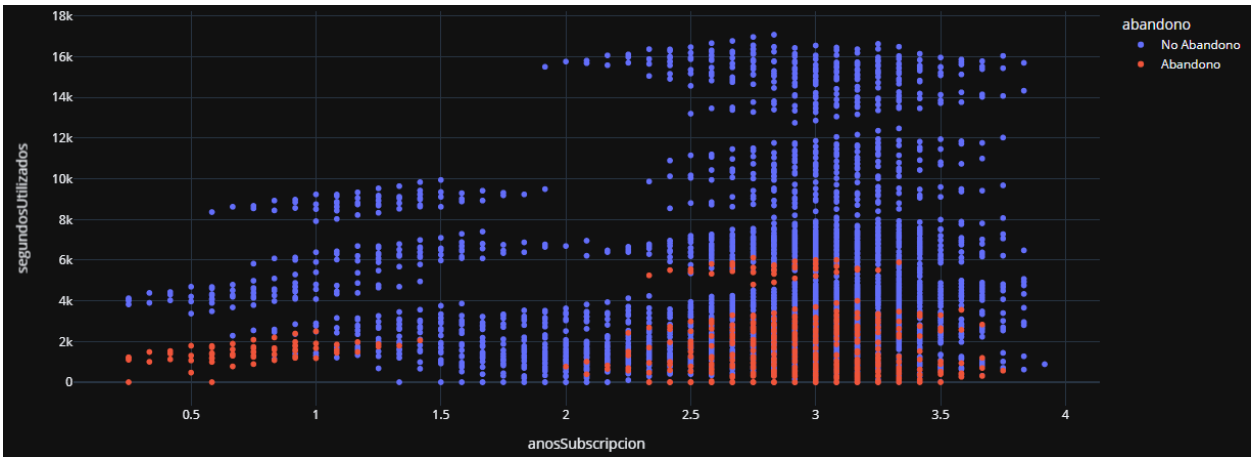
La frecuencia de uso inferior a 100 tiene alto impacto en el abandono

EJECUCION DE SCATTER EN TIEMPO DE SUBSCRIPCION Y USO DE SMS VS. ABANDONO



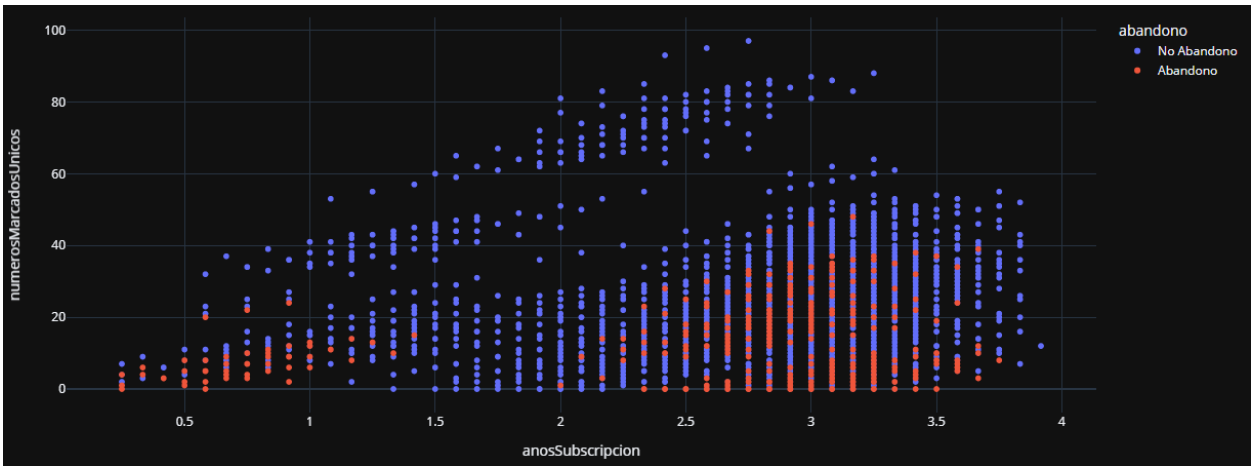
El uso de SMS bajo tiene alto impacto en el abandono.

EJECUCION DE SCATTER EN TIEMPO DE SUBSCRIPCION Y TIEMPO UTILIZADO VS. ABANDONO



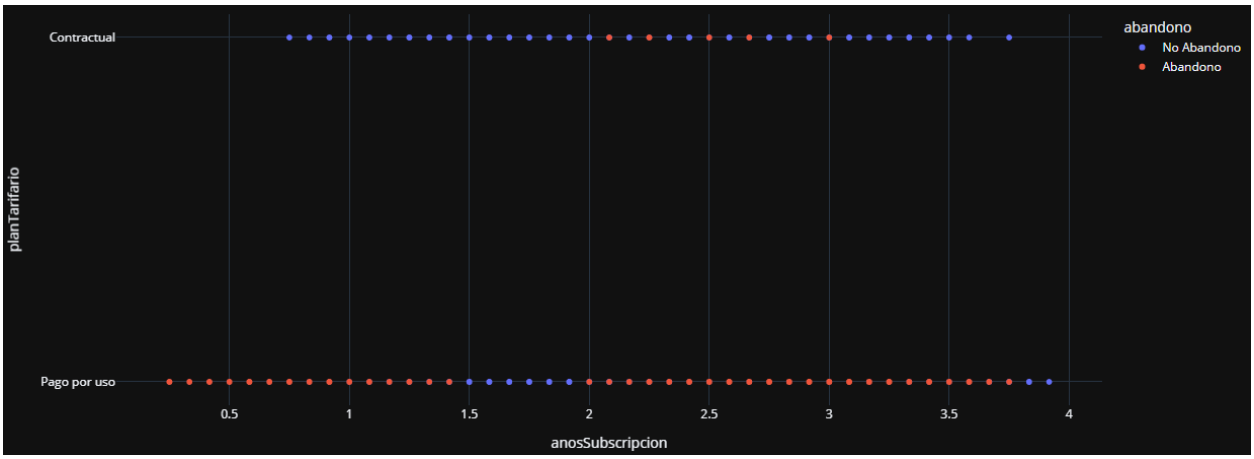
El bajo tiempo de utilización tiene un alto impacto en el abandono.

EJECUCION DE SCATTER EN TIEMPO DE SUBSCRIPCION Y NUMEROS UNICOS VS. ABANDONO

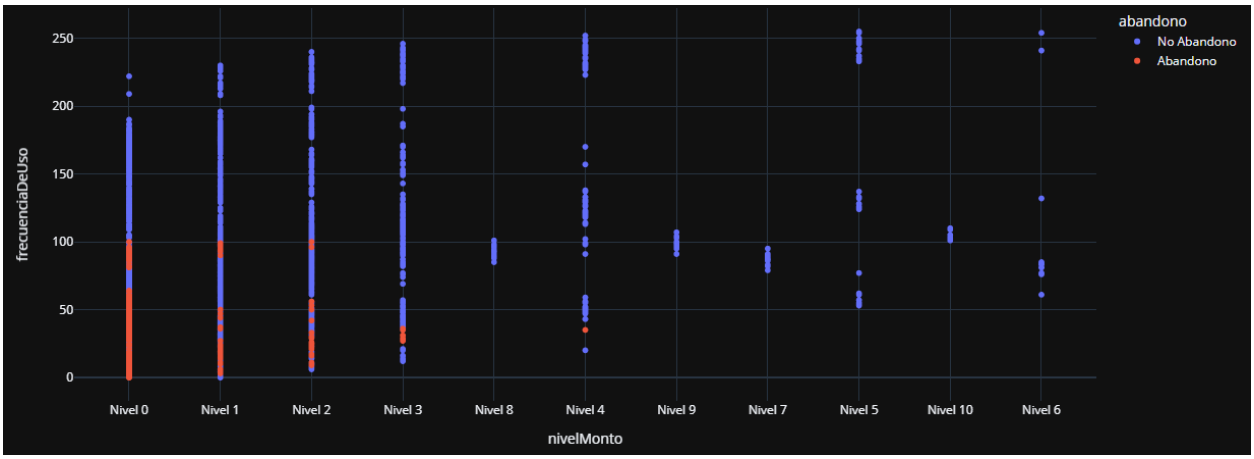


Los menores números de marcado tienen correlación con el abandono.

EJECUCION DE SCATTER EN TIEMPO DE SUBSCRIPCION Y PLAN TARIFARIO VS. ABANDONO



EJECUCION DE SCATTER EN NIVEL DE MONTO Y FRECUENCIA DE USO VS. ABANDONO



e) CONCLUSIONES DEL ANÁLISIS EXPLORATORIO

Se identifican las siguientes conclusiones e hipótesis:

- Los abandonos tienen una correlación con el estatus de los clientes y la cantidad de quejas.
- Sin embargo, las quejas y fallas se concentran en los usuarios que menos ingresan a la empresa.
- Ejemplos de esto son los usuarios que no tienen una cuenta con estatus activa, que tienen un plan de pago por uso y que no tienen un valor alto de Cliente. Estos son los que más abandonan.
- Las columnas de edad y grupo de edad tienen una distribución prácticamente idéntica, por lo que podemos prescindir de la columna edad.
- Los niveles de monto inferior son las que tienen más usuarios, pero también las que generan menor valor a la empresa, extrañamente en este segmento es donde más quejas y más abandono tenemos.
- A manera de hipótesis suponemos que a mayor nivel de monto mejora el servicio y por esto es que se diferencia la cantidad de quejas y abandono.
- La edad es también uno de los factores de abandono, la mayoría de los abandonos se identifican entre personas de edad mediana, haciendo una campana de gauss, ya que los mayores y los más jóvenes son las poblaciones de menor abandono.
- Las ventanas de abandono se dan principalmente en clientes nuevos (1 a 15 meses) y usuarios maduros (24 a 45 meses).
- La gran cantidad del abandono se da entre clientes con Nivel de monto 0 y 3.
- Las personas con mayor frecuencia de uso son las que menos abandonan (frecuencia de uso, minutos utilizados, uso de SMS, etc.).

f) SELECCIÓN DE CARACTERÍSTICAS

Previamente se había validado que no había columnas nulas, y aunque había algunas filas repetidas, hace sentido que esto pueda pasar por el volumen de datos.

Previamente a esta sección se habían generado variables nuevas para simplificar la lectura de datos:

- minutosUtilizados (float)– convierte segundos a minutos
- anosSubscripcion (float) – convierte meses a años

En el nuevo modelo de datos no utilizaremos las columnas siguientes:

- Edad – Ya existe Grupo de Edad que es más óptima para los modelos
- segundosUtilizados – ya tenemos minutosUtilizados
- mesesSubscripcion - ya tenemos anosSubscripcion

Se realizaron las siguientes actividades para simplificar los datos y que puedan ser procesados por los diferentes modelos:

- Generar copia del nuevo dataframe en df_final solo con las columnas necesarias
- Identificar los tipos de dato en cada variable
- Convertir las columnas categóricas a enteros y boleanos
- Reconvertir datos a enteros, float y object según corresponda
- Categorizar los datos entre objetos y no objetos
- Validar la conversión de datos
- Validamos que el sistema sigue identificando las etiquetas de las variables categóricas

iii. COMPARACIÓN DE MODELOS

Evaluaremos 10 modelos predictivos para identificar cual funciona mejora para predecir si nuestros clientes tienen riesgo de abandono con los datos que tenemos disponibles:

a) K-NEAREST NEIGHBORS (KNN)
b) DECISION TREE CLASSIFIER
c) RANDOM FOREST CLASSIFIER
d) ADA BOOST CLASSIFIER
e) GRADIENT BOOSTING CLASSIFIER
f) STOCHASTIC GRADIENT BOOSTING (SGB)
g) XGBOOST
h) CAT BOOST CLASSIFIER
i) EXTRA TREES CLASSIFIER
j) LGBM CLASSIFIER

Con algunos de estos modelos pudimos modificar algunas de los hiper parámetros y elegir configuraciones que optimicen los resultados. Mostraremos en este documento el desempeño general con la configuración estándar y para los modelos que lo permiten los ajustes que realizamos para optimizar junto con sus resultados.

a) K-NEAREST NEIGHBORS (KNN)

```
Training Accuracy of KNN is 0.9428571428571428
Test Accuracy of KNN is 0.8920634920634921

Confusion Matrix :-
[[745  31]
 [ 71  98]]

Classification Report :-
              precision    recall  f1-score   support

           0       0.91      0.96      0.94       776
           1       0.76      0.58      0.66       169

    accuracy                           0.89       945
   macro avg       0.84      0.77      0.80       945
weighted avg       0.89      0.89      0.89       945
```

b) DECISION TREE CLASSIFIER

```
Training Accuracy of Decision Tree Classifier is 0.9927437641723356
Test Accuracy of Decision Tree Classifier is 0.926984126984127

Confusion Matrix :-
[[746  30]
 [ 39 130]]

Classification Report :-
              precision    recall  f1-score   support

           0       0.95      0.96      0.96       776
           1       0.81      0.77      0.79       169

    accuracy                           0.93       945
   macro avg       0.88      0.87      0.87       945
weighted avg       0.93      0.93      0.93       945
```

```
Training Accuracy of Decision Tree Classifier is 0.9609977324263038
Test Accuracy of Decision Tree Classifier is 0.9216931216931217

Confusion Matrix :-
[[754  22]
 [ 52 117]]

Classification Report :-
              precision    recall  f1-score   support

           0       0.94      0.97      0.95       776
           1       0.84      0.69      0.76       169

    accuracy                           0.92       945
   macro avg       0.89      0.83      0.86       945
weighted avg       0.92      0.92      0.92       945
```

OPTIMIZACIÓN DEL DECISION TREE CLASSIFIER:

a. **`min_samples_split`** debería ser mayor que 1

`min_samples_split` debe ser al menos 2. Al configurar en 1, el árbol podría seguir dividiendo incluso cuando haya un solo ejemplo, lo que resulta en sobreajuste.

```
grid_param = {
    'criterion': ['gini', 'entropy'],
    'max_depth': [3, 5, 7, 10],
    'splitter': ['best', 'random'],
    'min_samples_leaf': [1, 2, 3, 5, 7],
    'min_samples_split': [2, 3, 5, 7],
    'max_features': ['auto', 'sqrt', 'log2'] }
```

b. Evaluación adicional de hiperparámetros:

Agregar `random_state` para asegurar la reproducibilidad.

```
dtc = DecisionTreeClassifier(random_state=42)
```

c. Uso de class_weight para manejar el desbalanceo:

Muchos modelos (como Random Forest, Decision Tree, etc.) pueden beneficiarse de la configuración de `class_weight='balanced'` para manejar el desbalanceo de clases.

```
dtc = DecisionTreeClassifier(class_weight='balanced', random_state=42)
```

d. Uso de RandomizedSearchCV en lugar de GridSearchCV:

`RandomizedSearchCV` puede ser más eficiente en cuanto a tiempo de computación cuando hay una gran cantidad de hiperparámetros, ya que busca en una cantidad aleatoria de combinaciones en lugar de probar todas las combinaciones posibles.

```
from sklearn.model_selection import RandomizedSearchCV
random_search_dtc = RandomizedSearchCV(dtc, grid_param, cv=5, n_iter=50, n_jobs=-1,
random_state=42, verbose=1)
random_search_dtc.fit(X_train, y_train)
# Best parameters and best score
print(random_search_dtc.best_params_)
print(random_search_dtc.best_score_)
```

e. Resultados con el ajuste:

Con la nueva configuración mejoramos algunos puntos para identificar el "No Abandono" sin embargo el modelo era mejor anteriormente, por lo que guardaremos esta configuración para en algún momento probarla con mayor cantidad de datos en conjuntos más desbalanceados.

```
Training Accuracy of Decision Tree Classifier is 0.9918367346938776
Test Accuracy of Decision Tree Classifier is 0.928042328042328

Confusion Matrix :-
[[744  32]
 [ 36 133]]

Classification Report :-
              precision    recall  f1-score   support

           0       0.95      0.96      0.96       776
           1       0.81      0.79      0.80       169

    accuracy                           0.93       945
   macro avg       0.88      0.87      0.88       945
weighted avg       0.93      0.93      0.93       945
```

```
Training Accuracy of Decision Tree Classifier is 0.9310657596371882
Test Accuracy of Decision Tree Classifier is 0.8793650793650793

Confusion Matrix :-
[[684  92]
 [ 22 147]]

Classification Report :-
              precision    recall  f1-score   support

           0       0.97      0.88      0.92       776
           1       0.62      0.87      0.72       169

    accuracy                           0.88       945
   macro avg       0.79      0.88      0.82       945
weighted avg       0.91      0.88      0.89       945
```

c) RANDOM FOREST CLASSIFIER

```
Training Accuracy of Random Forest Classifier is 0.9800453514739229
Test Accuracy of Random Forest Classifier is 0.944973544973545

Confusion Matrix :-
[[757  19]
 [ 33 136]]

Classification Report :-
              precision    recall  f1-score   support

           0       0.96      0.98      0.97       776
           1       0.88      0.80      0.84       169

    accuracy                           0.94       945
   macro avg       0.92      0.89      0.90       945
weighted avg       0.94      0.94      0.94       945
```

OPTIMIZACIÓN DEL RANDOM FOREST CLASSIFIER:

c. Uso de **class_weight** para manejar el desbalanceo:

- Muchos modelos (como Random Forest, Decision Tree, etc.) pueden beneficiarse de la configuración de `class_weight='balanced'` para manejar el desbalanceo de clases.

```
rd_clf = RandomForestClassifier(criterion='entropy', max_depth=11, max_features='auto',
                                min_samples_leaf=2, min_samples_split=3, n_estimators=130,
                                class_weight='balanced', random_state=42)
```

El modelo optimizado mejora para identificar los casos de No abandono, pero pierde puntos en abandono. Estos cambios definitivamente serán interesantes para probar con volúmenes de datos más grandes.

```
Training Accuracy of Random Forest Classifier is 0.980498866213152
Test Accuracy of Random Forest Classifier is 0.9396825396825397

Confusion Matrix :-
[[742  34]
 [ 23 146]]

Classification Report :-
              precision    recall  f1-score   support

           0       0.97      0.96      0.96       776
           1       0.81      0.86      0.84       169

    accuracy                           0.94       945
   macro avg       0.89      0.91      0.90       945
weighted avg       0.94      0.94      0.94       945
```

d) ADA BOOST CLASSIFIER

```
Training Accuracy of Ada Boost Classifier is 0.9927437641723356
Test Accuracy of Ada Boost Classifier is 0.9365079365079365

Confusion Matrix :-
[[756  20]
 [ 40 129]]

Classification Report :-
              precision    recall  f1-score   support

           0       0.95      0.97      0.96       776
           1       0.87      0.76      0.81       169

    accuracy                           0.94       945
   macro avg       0.91      0.87      0.89       945
weighted avg       0.93      0.94      0.93       945
```

OPTIMIZACIÓN DEL ADA BOOST CLASSIFIER:

Mejora del ajuste de hiperparámetros para los modelos de Boosting:

Incluir ajustes de hiperparámetros para modelos como AdaBoost, Gradient Boosting, y XGBoost, ya que esto podría mejorar significativamente su rendimiento.

```
# AdaBoost
ada = AdaBoostClassifier(base_estimator=dtc, n_estimators=100, learning_rate=0.1,
random_state=42)
```

Se detectan ligeras mejoras al detectar el Abandono con los nuevos parámetros.

```
Training Accuracy of Ada Boost Classifier is 0.9927437641723356
Test Accuracy of Ada Boost Classifier is 0.9365079365079365

Confusion Matrix :-
[[762  14]
 [ 46 123]]

Classification Report :-
              precision    recall  f1-score   support

           0       0.94      0.98      0.96       776
           1       0.90      0.73      0.80       169

    accuracy                           0.94       945
   macro avg       0.92      0.85      0.88       945
weighted avg       0.93      0.94      0.93       945
```

e) GRADIENT BOOSTING CLASSIFIER

```
Training Accuracy of Gradient Boosting Classifier is 0.9705215419501134
Test Accuracy of Gradient Boosting Classifier is 0.9386243386243386

Confusion Matrix :-
[[754  22]
 [ 36 133]]

Classification Report :-
              precision    recall  f1-score   support

           0       0.95      0.97      0.96       776
           1       0.86      0.79      0.82       169

    accuracy                           0.94       945
   macro avg       0.91      0.88      0.89       945
weighted avg       0.94      0.94      0.94       945
```

Inicialmente, el modelo fue entrenado con sus hiperparámetros predeterminados, lo que resultó en un buen rendimiento, pero dejó margen para la optimización. Implementé un ajuste de hiperparámetros utilizando GridSearchCV, lo que permitió probar diversas combinaciones de parámetros como n_estimators, learning_rate, max_depth, entre otros. A través de este proceso, se identificó que los siguientes valores proporcionaban un mejor equilibrio entre precisión y capacidad de generalización:

n_estimators: Incrementar el número de estimadores a 200 mejoró ligeramente la precisión sin causar un sobreajuste significativo. 'n_estimators': [100, 200, 300]

learning_rate: Reducir la tasa de aprendizaje a 0.05 permitió que el modelo hiciera ajustes más refinados, lo que mejoró su capacidad para capturar patrones más complejos en los datos sin sobreajustarse. 'learning_rate': [0.01, 0.05, 0.1]

max_depth: Ajustar la profundidad máxima de los árboles a 5 ayudó a capturar relaciones más profundas sin hacer el modelo excesivamente complejo. 'max_depth': [3, 4, 5, 6]

subsample: Utilizar un subsample de 0.9 proporcionó una regularización adicional, ayudando a evitar el sobreajuste y mejorando la generalización del modelo en datos no vistos. 'subsample': [0.8, 0.9, 1.0]

```
Fitting 5 folds for each of 2916 candidates, totalling 14580 fits
Training Accuracy of Gradient Boosting Classifier is 0.9913832199546485
Test Accuracy of Gradient Boosting Classifier is 0.9407407407407408

Confusion Matrix :-
[[761  15]
 [ 41 128]]

Classification Report :-
              precision    recall  f1-score   support

           0       0.95      0.98      0.96       776
           1       0.90      0.76      0.82       169

    accuracy                           0.94       945
   macro avg       0.92      0.87      0.89       945
weighted avg       0.94      0.94      0.94       945
```

f) STOCHASTIC GRADIENT BOOSTING (SGB)

```
Training Accuracy of Stochastic Gradient Boosting is 0.9918367346938776
Test Accuracy of Stochastic Gradient Boosting is 0.9375661375661376

Confusion Matrix :-
[[755  21]
 [ 38 131]]

Classification Report :-
              precision    recall  f1-score   support

           0       0.95      0.97      0.96       776
           1       0.86      0.78      0.82       169

    accuracy                           0.94       945
   macro avg       0.91      0.87      0.89       945
weighted avg       0.94      0.94      0.94       945
```

g) XGBOOST

```
Training Accuracy of XgBoost is 0.9927437641723356
Test Accuracy of XgBoost is 0.9417989417989417

Confusion Matrix :-
[[758  18]
 [ 37 132]]

Classification Report :-
              precision    recall  f1-score   support

           0       0.95      0.98      0.96       776
           1       0.88      0.78      0.83       169

    accuracy                           0.94       945
   macro avg       0.92      0.88      0.90       945
weighted avg       0.94      0.94      0.94       945
```

OPTIMIZACIÓN DEL XGBOOST:

Mejora del ajuste de hiperparámetros para los modelos de Boosting:

Incluir ajustes de hiperparámetros para modelos como AdaBoost, Gradient Boosting, y XGBoost, ya que esto podría mejorar significativamente su rendimiento.

XGBoost (puedes afinar más parámetros como gamma, subsample, colsample_bytree, etc.)

```
xgb = XGBClassifier(objective='binary:logistic', learning_rate=0.05, max_depth=4, n_estimators=200, random_state=42)
```

El modelo identifica mejoras menores con el ajuste para datos de prueba.

```
Training Accuracy of XgBoost is 0.9755102040816327
Test Accuracy of XgBoost is 0.944973544973545

Confusion Matrix :-
[[757  19]
 [ 33 136]]

Classification Report :-
              precision    recall  f1-score   support

           0       0.96      0.98      0.97       776
           1       0.88      0.80      0.84       169

    accuracy                           0.94       945
   macro avg       0.92      0.89      0.90       945
weighted avg       0.94      0.94      0.94       945
```

h) CAT BOOST CLASSIFIER

```
Training Accuracy of Cat Boost Classifier is 0.946031746031746
Test Accuracy of Cat Boost Classifier is 0.9195767195767196

Confusion Matrix :-
[[764  12]
 [ 64 105]]

Classification Report :-
              precision    recall  f1-score   support

           0       0.92      0.98      0.95       776
           1       0.90      0.62      0.73       169

    accuracy                           0.92       945
   macro avg       0.91      0.80      0.84       945
weighted avg       0.92      0.92      0.91       945
```

OPTIMIZACIÓN DEL CAT BOOST CLASSIFIER:

Manejo de Iteraciones en CatBoost:

El número de iteraciones en `CatBoostClassifier` parece bajo (10). Se incrementará para permitir un mejor ajuste del modelo.

```
cat = CatBoostClassifier(iterations=500, learning_rate=0.1, depth=6, random_state=42, verbose=100)
```

```
Training Accuracy of Cat Boost Classifier is 0.9927437641723356
Test Accuracy of Cat Boost Classifier is 0.944973544973545

Confusion Matrix :-
[[760  16]
 [ 36 133]]

Classification Report :-
              precision    recall  f1-score   support

           0       0.95      0.98      0.97       776
           1       0.89      0.79      0.84       169

    accuracy                           0.94       945
   macro avg       0.92      0.88      0.90       945
weighted avg       0.94      0.94      0.94       945
```

i) EXTRA TREES CLASSIFIER

```
Training Accuracy of Extra Trees Classifier is 0.9927437641723356
Test Accuracy of Extra Trees Classifier is 0.9513227513227513

Confusion Matrix :-
[[766  10]
 [ 36 133]]

Classification Report :-
              precision    recall  f1-score   support

           0       0.96      0.99      0.97       776
           1       0.93      0.79      0.85       169

    accuracy                           0.95       945
   macro avg       0.94      0.89      0.91       945
weighted avg       0.95      0.95      0.95       945
```

j) LGBM CLASSIFIER

```
Training Accuracy of LGBM Classifier is 0.9927437641723356
Test Accuracy of LGBM Classifier is 0.9407407407407408

[[755  21]
 [ 35 134]]

              precision    recall  f1-score   support

           0       0.96      0.97      0.96       776
           1       0.86      0.79      0.83       169

    accuracy                           0.94       945
   macro avg       0.91      0.88      0.90       945
weighted avg       0.94      0.94      0.94       945
```

k) SELECCIÓN DEL MODELO GANADOR

Al analizar los resultados de los modelos elegidos, se puede observar que varios de ellos muestran un rendimiento fuerte en términos de precisión (accuracy) y otras métricas de evaluación.

	Model	Score
8	Extra Trees Classifier	0.954497
6	XgBoost	0.944974
7	Cat Boost	0.944974
4	Gradient Boosting Classifier	0.940741
2	Random Forest Classifier	0.939683
3	Ada Boost Classifier	0.938624
5	Stochastic Gradient Boosting	0.938624
1	Decision Tree Classifier	0.920635
0	KNN	0.892063

Se muestra a continuación un resumen de los modelos y recomendaciones sobre cuál sería más adecuado para ser considerado como el mejor modelo:

ANÁLISIS DE CADA MODELO:

1. Extra Trees Classifier

- **Test Accuracy**: 0.954
- **Training Accuracy**: 0.992
- **Observación**: Este modelo tiene el mejor rendimiento general en términos de precisión y F1-Score. Sin embargo, la alta precisión en el conjunto de entrenamiento podría indicar un posible sobreajuste.

2. XGBoost

- **Test Accuracy**: 0.944
- **Training Accuracy**: 0.976
- **Observación**: XGBoost es robusto y ofrece un excelente equilibrio entre precisión, recall y F1-Score, especialmente para la clase minoritaria.

3. CatBoost Classifier

- **Test Accuracy**: 0.944
- **Training Accuracy**: 0.992

- **Observación**: CatBoost también ofrece un rendimiento muy similar a XGBoost con una alta precisión en prueba. Es conocido por su eficiencia en el manejo de características categóricas y su capacidad para manejar datos desbalanceados.

4. Random Forest Classifier

- **Test Accuracy**: 0.939
- **Training Accuracy**: 0.980
- **Observación**: Random Forest es confiable y proporciona un buen rendimiento general. Tiene un buen equilibrio entre precisión y recall, y es menos propenso al sobreajuste en comparación con Extra Trees.

5. Gradient Boosting Classifier

- **Test Accuracy**: 0.940 (en la imagen adjunta)
- **Training Accuracy**: 0.991
- **Observación**: Gradient Boosting es efectivo, especialmente cuando se ajustan los hiperparámetros. Ofrece un rendimiento sólido con una buena capacidad de generalización, similar a Random Forest.

6. Stochastic Gradient Boosting (SGB)

- **Test Accuracy**: 0.938
- **Training Accuracy**: 0.992
- **Observación**: SGB es robusto y tiene un rendimiento comparable a AdaBoost, pero con un ligero sesgo hacia la clase mayoritaria. Funciona bien cuando hay un cierto nivel de ruido en los datos.

7. AdaBoost Classifier

- **Test Accuracy**: 0.938
- **Training Accuracy**: 0.992
- **Observación**: AdaBoost es simple y efectivo para mejorar modelos base. Sin embargo, tiene un rendimiento ligeramente inferior en la clase minoritaria en comparación con los modelos anteriores.

8. LGBM Classifier

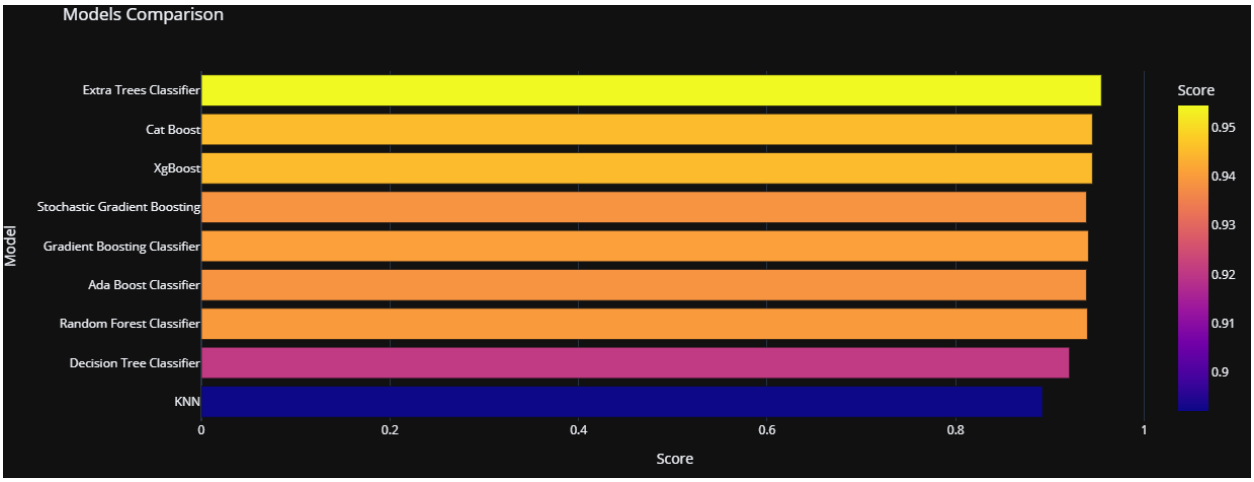
- **Test Accuracy**: 0.940
- **Training Accuracy**: 0.992
- **Observación**: LGBM es rápido y eficiente, especialmente con grandes conjuntos de datos. Ofrece un rendimiento similar a Gradient Boosting, pero con ligeras diferencias en precisión y recall.

9. Decision Tree Classifier

- **Test Accuracy**: 0.920
- **Training Accuracy**: 0.970
- **Observación**: Decision Tree tiene un rendimiento decente, pero muestra signos de sobreajuste debido a la alta precisión en el conjunto de entrenamiento en comparación con el conjunto de prueba.

10. K-Nearest Neighbors (KNN)

- **Test Accuracy**: 0.892
- **Training Accuracy**: 0.942
- **Observación**: KNN tiene la precisión más baja entre los modelos probados. Es simple, pero no tan efectivo en comparación con los modelos de ensemble más avanzados.



CONCLUSIÓN DE LOS RESULTADOS:

El **Extra Trees Classifier** es el modelo con el mejor rendimiento, seguido de cerca por **XGBoost** y **CatBoost**. Estos modelos ofrecen un excelente equilibrio entre precisión, recall y robustez general. Por otro lado, **KNN** y **Decision Tree** están en el extremo inferior del rendimiento, siendo menos efectivos para este conjunto de datos en comparación con los modelos de boosting y ensemble más sofisticados.

Este orden proporciona una visión clara de cuáles modelos son más efectivos y cuáles podrían necesitar ajustes adicionales o ser reemplazados por alternativas más robustas.

RECOMENDACIONES:

Con base en los resultados presentados:

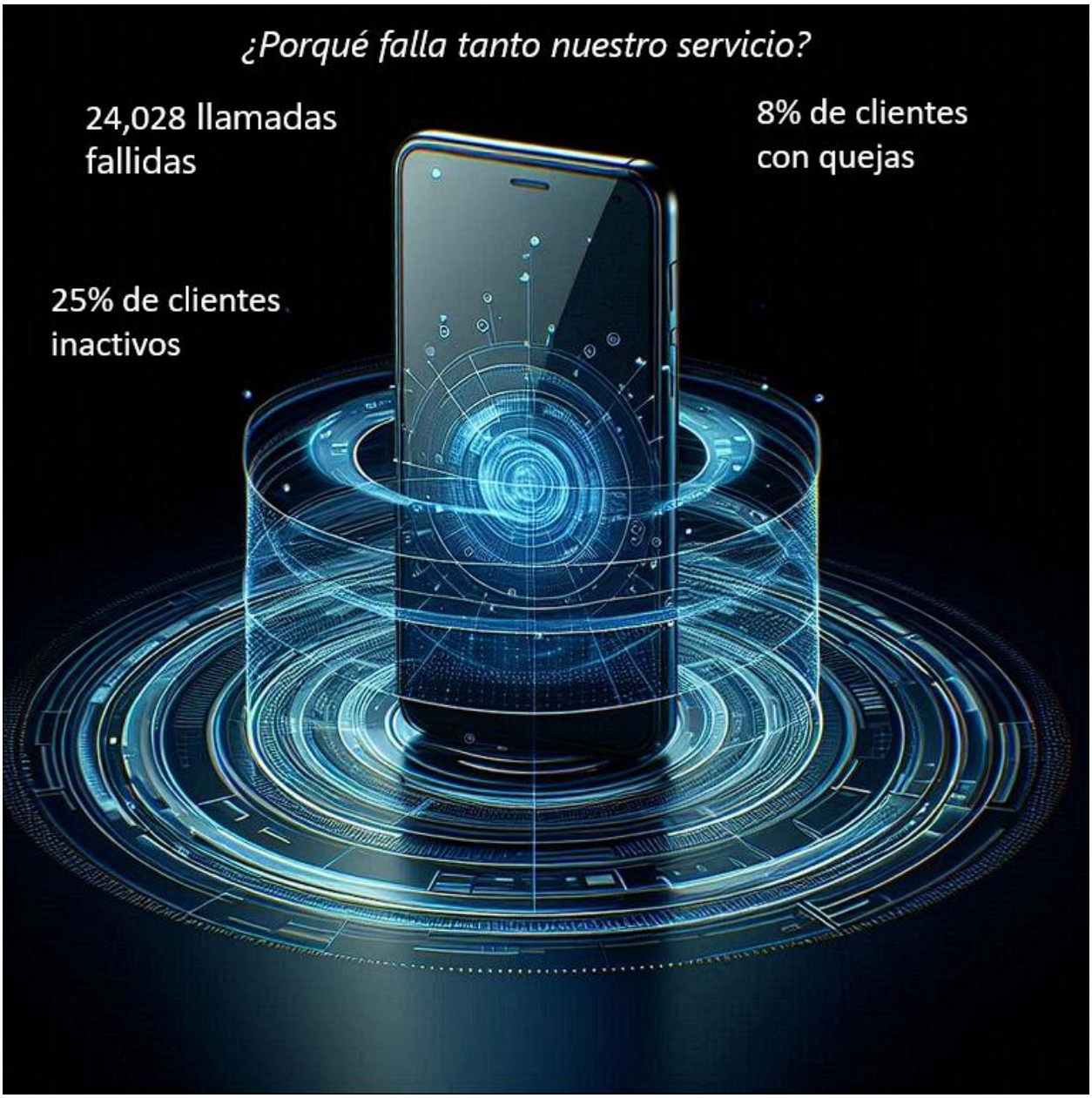
- **Extra Trees Classifier** es el modelo con la mejor precisión en prueba (0.954), junto con un excelente F1-Score y recall. Es un excelente candidato para ser el modelo seleccionado, especialmente si se puede mitigar el riesgo de sobreajuste.
- XGBoost y CatBoost son otros modelos destacados, ambos con precisiones en prueba de 0.944, lo que demuestra que son muy competitivos. Además, estos modelos son conocidos por su robustez y capacidad para manejar datos desbalanceados, lo que los convierte en una elección excelente si buscas un modelo equilibrado que maneje bien la clase minoritaria.

iv. IMPLICACIONES DEL NEGOCIO

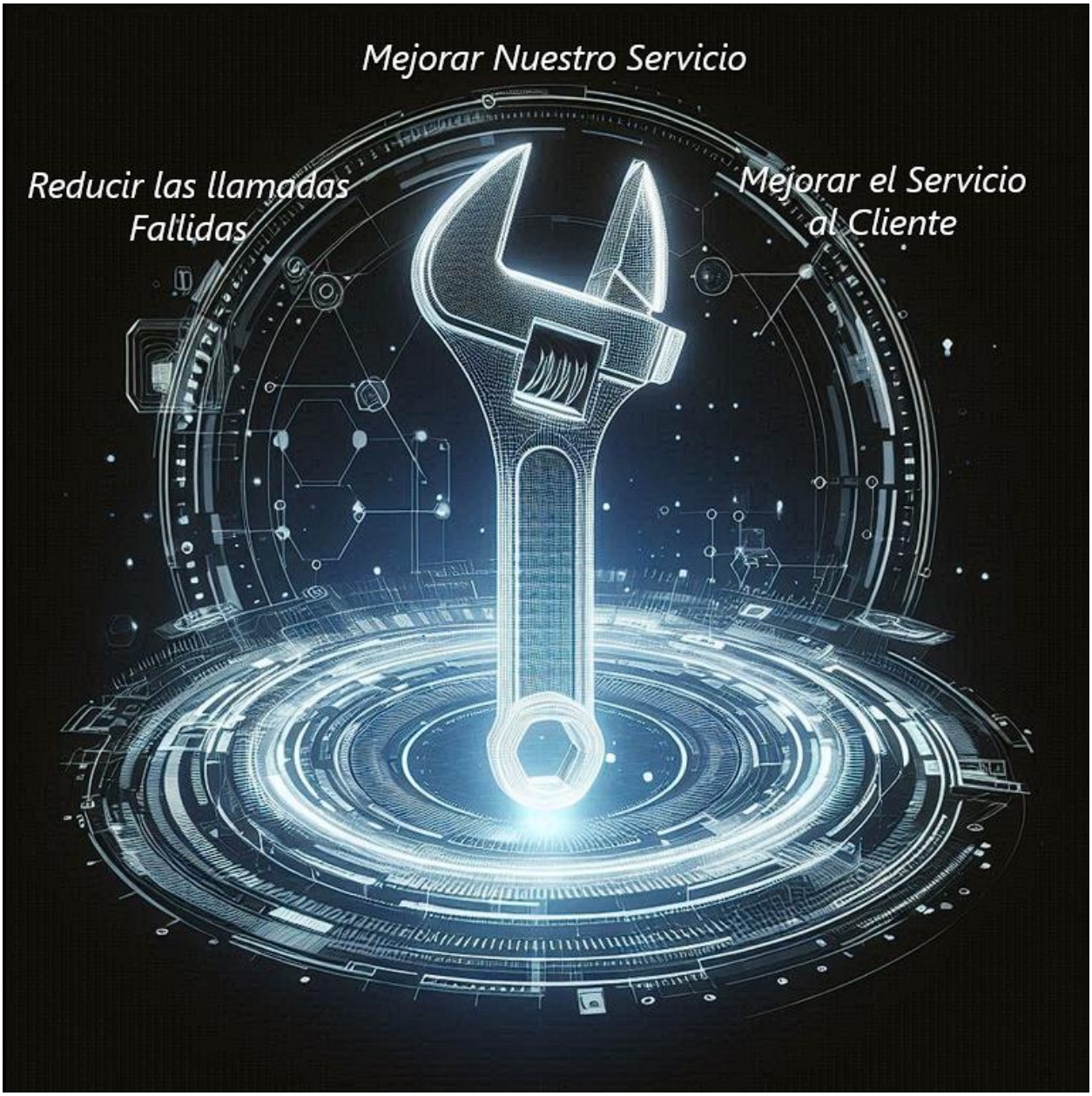
a) RETOS DEL NEGOCIO

- Conocer a nuestro cliente y porque el 20% está inconforme de manera continua.

- Identificar porque nuestro Servicio presenta tantas fallas y porque sucede principalmente entre nuestros usuarios de menor monto ingresado.



- Crear una estrategia que permita reducir las fallas en nuestro servicio.



- Cambiar la imagen negativa que se ha dado de la empresa, creando una estrategia de Marketing para que los clientes renueven por más tiempo y se reduzca la inactividad de algunas cuentas, consideremos

b) SOLUCIÓN PROPUESTA

1. Validación de datos de abandono del Cliente.

PROPUESTA:

- Realización de un estudio de mercado enfocado en clientes que han abandonado.
    a) Población objetivo: usuarios entre 25 a 54 años, que fueron nuestros clientes en plan de pago por uso, con antigüedad de 0 a 3 años.
    b) Objetivo del estudio: Identificar las causas de abandono, los servicios que gusta usar, las causas de la poca actividad, las promociones que le gustaría disfrutar.

2. Validación de fallas técnicas.

PROPUESTA:

- Realizar un estudio técnico para validar porque se dan las fallas en las llamadas.
- Validar la posibilidad de compensar a los clientes que han tenido fallas en el servicio.
- Validar si tenemos la posibilidad de dar servicio a todos los clientes que tenemos.

3. Mejorar nuestro servicio

PROPUESTA:

- En base a los resultados del estudio anterior, validar la posibilidad de realizar las correcciones: mejorar tecnología actual, contratar más personal para monitoreo, reparaciones, atención al cliente.
- Aplicar estrategias de mejora continua, alimentando el modelo de manera recurrente para irlo mejorando y tener en el radar posibles abandonos próximos.

4. Campañas de Marketing

PROPUESTA:

- Una vez corregidas las fallas en el servicio, promover que nuestro servicio ha mejorado por las nuevas inversiones realizadas.
- Realizar promociones o planes de fidelidad en los que se brinden beneficios a los usuarios que conserven su servicio uno, dos o tres años, así mientras más tiempo se queden mejores precios podrán obtener. Ej. más SMS por prepagos realizados.
- Reactivar a los usuarios inactivos, ofreciendo precios especiales en llamadas nocturnas.
- Ofrecer promociones para clientes con pocos contactos, como precios especiales para hablar con tus personas favoritas de manera recurrente.

c) BENEFICIOS DE NEGOCIO

1. Validación de datos de abandono del Cliente.
BENEFICIOS:

- Validar nuestra hipótesis para seguir utilizándolo con confianza
- Utilizar la nueva data que nos comenten para generar un modelo que considere otras variables no identificadas hasta ahora.

2. Validación de fallas técnicas.
BENEFICIOS:

- Identificar las fallas actuales y tener un plan de acción para eliminarlas o en su defecto controlarlas.
- Tener claro donde debe invertirse para mejorar el servicio y garantizar el ancho de banda para poder seguir ofreciendo junto nuestros clientes.

3. Mejorar nuestro servicio
BENEFICIOS:

- Existe una correlación de abandono por las quejas de usuarios, y el mal servicio, realizar las correcciones en el servicio es importante para mantenernos competitivos y reducir el abandono.
- Seguir utilizando los modelos predictivos e incorporando nuevos procesos nos garantizaran poder tener cierto grado de control y previsión rumbo a una mejora continua.

Campañas de Marketing
BENEFICIOS:

- Dar confianza a clientes que abandonaron y clientes nuevos que nuestro servicio ha mejorado y que nos haremos responsables de las fallas.
- Contar con un mecanismo de compensación a los clientes con fallas permitirá resarcir el daño a clientes actuales y reducir la mala imagen actual.
- Las campañas de fidelidad ayudarán a reducir las tendencias de abandono de clientes que abandonan al año o a los dos y tres años respectivamente.
- Promociones de más tiempo aire para clientes con plan de uso, ayudará a reducir la falta de uso en horarios donde el servicio no se ocupa.
- Los clientes con pocos contactos son población que abandona, promociones para ellos nos ayudarán a conservarlos.