Proyecto de Introducción a la Inteligencia Artificial

Entrega Final

Cristian Camilo Julio Mejía

Ingeniería de Sistemas

Universidad de Antioquia

Introducción a la Inteligencia Artificial

Raúl Ramos Pollan

Noviembre

2023

Introducción

Cuando nos hablan de inteligencia artificial y no estamos muy relacionado con dicho concepto, podríamos llegar a pensar que es "algo" que resulta por arte de magia; procesos o tareas que se realizan de manea autónoma, se hacen solas con tan solo frotar una "barita mágica", pero no es así. Cuando nos vamos introduciendo un poco más en este tema, observamos que todo tiene su razón de ser, que no es un truco de magia, ni mucho menos, actos paranormales. Son procedimientos matemáticos que con la ayuda de algoritmos dan solución a un problema planteado. La estadística, la aritmética, la geometría, son algunas de las ramas matemáticas que se utilizan para procesar la información; mientras tanto, los algoritmos emplean modelos que a medida que se les proporciona grandes cantidades de datos estos van "aprendiendo" en el transcurso del tiempo. Es por ello, que en este proyecto académico se realizarán operaciones matemáticas básicas para procesar la información suministrada por una base de datos de telecomunicaciones, en donde se busca predecir si un cliente abandonará o no la compañía según características establecidas evaluándolas a través de modelos supervisados y no supervisados que indicarán si dichas predicciones son adecuadas por medio de curvas de aprendizaje.

Por otro lado, cabe mencionar que el proyecto está dividiendo en cuatro (4) notebook. En el primero, se realizó una exploración inicial de los datos, en donde se cargó la base de datos, se visualizó para revisar con que información se contaba (Fig.1.0) y se realizaron algunas gráficas en donde se resumió de manera adecuada la información (Fig.2.0).

```
Index(['CustomerID', 'Count', 'Country', 'State', 'City', 'Zip Code',
       'Lat Long', 'Latitude', 'Longitude', 'Gender', 'Senior Citizen',
       'Partner', 'Dependents', 'Tenure Months', 'Phone Service',
       'Multiple Lines', 'Internet Service', 'Online Security',
       'Online Backup', 'Device Protection', 'Tech Support', 'Streaming TV',
       'Streaming Movies', 'Contract', 'Paperless Billing', 'Payment Method',
       'Monthly Charges', 'Total Charges', 'Churn Label', 'Churn Value',
       'Churn Score', 'CLTV', 'Churn Reason'],
      dtype='object')

 (7043, 33)
```

Fig. 1.0. Descripción Inicial de las columnas

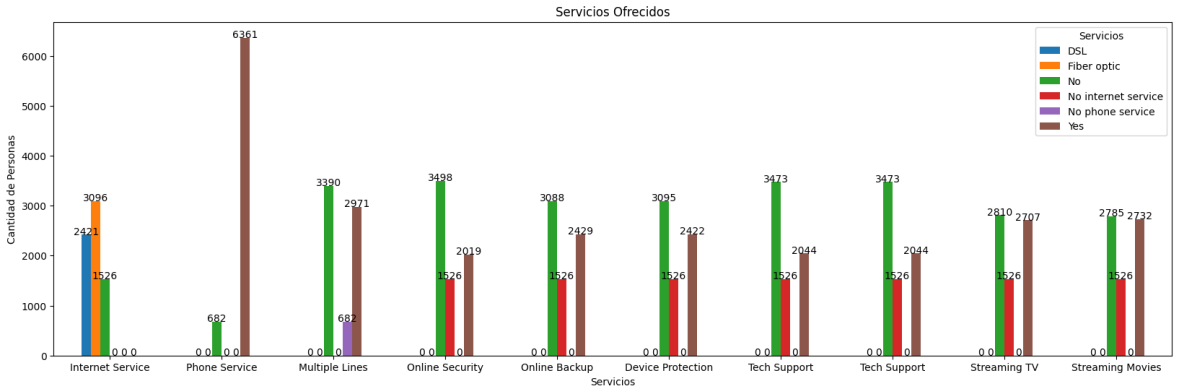

Fig. 2.0. Gráfica de Barras Servicios Ofrecidos Vs Cliente

En el segundo archivo se estableció la variable objetivo con la cual se construyó una gráfica de densidad en donde se visualizó la concentración de los clientes que permanecieron y no en la compañía (PROYECTO_ENTREGA2, pág. 2). A demás se agregó el porcentaje de valores faltantes y posteriormente se simularon (Fig. 3.0). También se normalizaron las variables numéricas para prevenir o disminuir datos atípicos estableciéndose en una misma escala (Fig. 4.0) y se realizó un primer modelo de Regresión logística (Fig. 5.0).

	ID	Valores Faltantes	Porcentaje Faltantes (%)
CLTV		0	0.000000
Churn Label		0	0.000000
Churn Reason		5174	73.463013
Churn Score		0	0.000000
Churn Value		0	0.000000
City		0	0.000000

Fig. 3.0. Porcentaje de Valores Faltantes

```
# Creando el normalizador
scaler = MinMaxScaler()

# Seleccionando solo las columnas numéricas para normalizar
num_cols = data_2.select_dtypes(include=[np.number]).columns

# Normalizando las columnas numéricas
data_2[num_cols] = scaler.fit_transform(data_2[num_cols])

# Imprimiendo los resultados
print(data_2)

```

Fig. 4.0. Normalizando Valores Numéricas

```
# Seleccionando las características y la variable objetivo
X = data_2[['Churn Score', 'CLTV']]
y = data_2['Churn Value']

# Dividiendo los datos en conjuntos de entrenamiento y prueba
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

```

```
# Creando el modelo
model = LogisticRegression()
```

Fig. 5.0. Implementación de un Primer Modelo (Regresión Logística)

En el tercer Notebook, a demás del modelo de Regresión Logística, también se empleó el de búsqueda aleatoria (Random Forest) se eligieron como modelos supervisados de estudio. En ambos casos se identificaron los mejores hiperparámetros y se evaluaron según la exactitud, precisión, sensibilidad, los valores de F1 y ROC respectivamente (Fig. 7.0).

```
Mejores parámetros para la regresión logística: {'C': 0.1}
Los mejores hiperparámetros son: {'n_estimators': 400, 'max_depth': 10}
```

Fig. 6.0. Hiperparámetros para ambos modelos

```
Precisión del modelo: 0.850958126330731
Sensibilidad del modelo: 0.6625
Precisión del modelo: 0.7794117647058824
F1 del modelo: 0.7162162162162162
ROC del modelo: 0.7940844895936571
```

```
Precisión del modelo: 0.8523775727466288
Sensibilidad del modelo: 0.5775
Precisión del modelo: 0.8555555555555555
F1 del modelo: 0.6895522388059702
ROC del modelo: 0.7694239345887017
```

Fig. 7.0. Evaluación de los modelos (Regresión y Random Forest)

En el cuarto cuaderno, se anexaron los modelos no supervisados como K-Means y DBSCAN; ambos son de agrupación en donde se busca es observar la cohesión y concentración de los datos, y no realizar predicciones como con los modelos supervisados (Fig. 8.0 y Fig. 9.0).

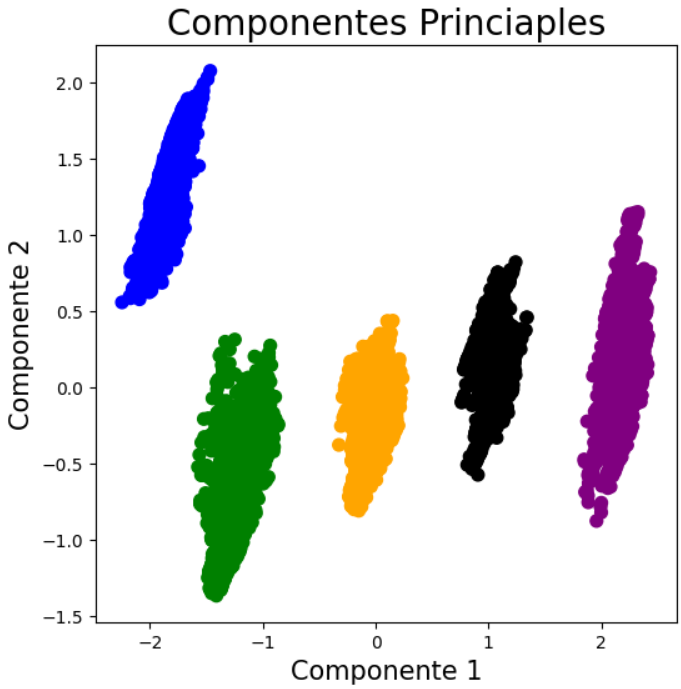

Fig. 8.0. Agrupación de las 5 primeras características según K-Means

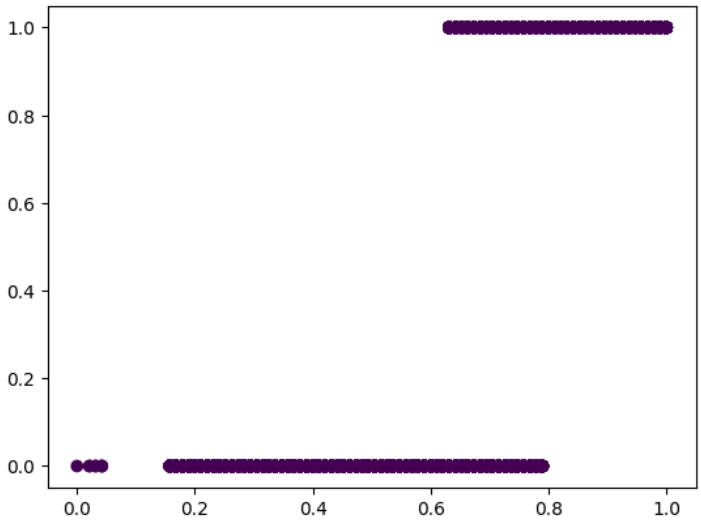

Fig. 9.0. Agrupación del Valor de Abandono (3) Vs al Puntaje de Abandono (2) según DBSCAN

***Nota**: Tanto el valor como el puntaje de abandono hacen parte del Dataset data_3, características tres (3) y dos (2) respectivamente.*

Descripción del DataSet

La base de dato a analizar fue adquirida a través de un Dataset alojado en Kaggle (Telco customer churn: IBM dataset), que tiene 7043 filas y 33 columnas. Este conjunto de datos contiene información sobre los clientes de una empresa de telecomunicaciones, incluida su información demográfica, los servicios a los que se han suscrito y si han abandonado o no. El documento que contiene los datos es un archivo en Excel. La información es la siguiente:

- CustomerID: ID único que identifica a cada cliente.
- Gender: El género del cliente: Masculino, Femenino
- Senior Citizen: Indica si el cliente tiene 65 años o más: Sí, No
- Tenure Months: Indica la cantidad total de meses que el cliente ha estado con la empresa al final del trimestre especificado anteriormente.
- Phone Service: Indica si el cliente contrata el servicio de telefonía residencial con la empresa: Sí, No
- Multiple Lines: Indica si el cliente contrata múltiples líneas telefónicas con la empresa: Sí, No
- Internet Service: Indica si el cliente contrata servicio de Internet con la empresa: No, DSL, Fibra Óptica, Cable.
- Contract: Indica el tipo de contrato actual del cliente: Mes a Mes, Un Año, Dos Años.
- Churn Label: Sí = el cliente dejó la empresa este trimestre. No = el cliente permaneció en la empresa. Directamente relacionado con el valor de abandono.
- Churn Value: 1 = el cliente dejó la empresa este trimestre. 0 = el cliente permaneció en la empresa. Directamente relacionado con Churn Label.
- Churn Score: un valor de 0 a 100 que se calcula utilizando la herramienta predictiva IBM SPSS Modeler. El modelo incorpora múltiples factores que se sabe que causan deserción. Cuanto mayor sea la puntuación, más probabilidades habrá de que el cliente abandone.
- CLTV: Valor de vida del cliente. Un CLTV previsto se calcula utilizando fórmulas corporativas y datos existentes. Cuanto mayor sea el valor, más valioso será el cliente. Se debe monitorear la deserción de los clientes de alto valor.
- Churn Reason: motivo específico de un cliente para abandonar la empresa. Directamente relacionado con la categoría de abandono.

Y otras características, como el lugar de residencia del cliente, la ciudad, el país, si tiene a alguien a cargo, el tipo de servicio elegido, etc.

Iteraciones de desarrollo

En este apartado básicamente solo se utilizará el Notebook cuarto (Modelos No Supervisados), porque este contine la implementación completa

Adquisición de los datos

```
[3]  1 import pandas as pd
     2
     3 data = pd.read_excel('/content/drive/MyDrive/UDEA/2023-02/IntroduccionIA/Proyecto/Telco_customer_churn.xlsx')
     4
```

```
1 import pandas as pd
2
3 # Cargar el archivo Excel en un data_2Frame
4 data = pd.read_excel('/content/Telco_customer_churn.xlsx')
```

Nota: Los datos de la Base de Datos Provienen de : https://www.kaggle.com/datasets/yeanzc/telco-customer-churn-ibm-dataset

Fig. 10. Cargando Base de datos desde Drive y localmente, respectivamente

Revisión inicial

```
1 data.head()
```

	CustomerID	Count	Country	State	City	Zip Code	Lat Long	Latitude	Longitude	Gender	...	Contract	Paperless Billing
0	3668-QPYBK	1	United States	California	Los Angeles	90003	33.964131, -118.272783	33.964131	-118.272783	Male	...	Month-to-month	Yes

Fig. 11. Observando la primera fila de la base de datos (data)

Excluyendo algunas columnas

```
1 columnas_a_excluir = ['CustomerID', 'Count', 'Country', 'State', 'Zip Code', 'Lat Long', 'Latitude', 'Longitude',...]
2
3 # Selecciona solo las columnas que no están en la lista de columnas a excluir
4 data_2= data[data.columns.difference(columnas_a_excluir)]
5
6 # Imprime el dataFrame filtrado
7 print(data_2)
```

Fig. 12. Imprimiendo base de datos filtrada (data_2)

Generando Modelos Supervisados

```python
# Seleccionando las características y la variable objetivo
X = data_2[['Churn Score', 'CLTV']]
y = data_2['Churn Value']

# Dividiendo los datos en conjuntos de entrenamiento y prueba
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state
```

```python
# Creando el modelo
model = LogisticRegression()
```

```python
# Entrenando el modelo
model.fit(X_train, y_train)
```

Fig. 13. Modelo de Regresión Logística

```python
# Seleccionando las características y la variable objetivo
X = data_2[['Churn Score', 'CLTV', 'Gender']]
y = data_2['Churn Value']

# Dividiendo los datos en conjuntos de entrenamiento y prueba
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Definiendo los hiperparámetros para la búsqueda aleatoria
hyperparameters = {'n_estimators': [100, 200, 300, 400, 500], 'max_depth': [None, 10, 20, 30, 40, 50]}

# Creando el modelo
model = RandomForestClassifier(n_estimators=100, random_state=42)

# Creando el objeto de búsqueda aleatoria
random_search = RandomizedSearchCV(model, hyperparameters, random_state=42)

# Entrenando el modelo
model.fit(X_train, y_train)

# Haciendo predicciones
y_pred = model.predict(X_test)
```

Fig. 14. Modelo Random Forest

Incluyendo solo las columnas numéricas

```
data_3 = data_2.select_dtypes(include=[np.number])

display(data_3)
```

Fig. 15. Base de datos con tan solo las características numéricas (data_3)

Generando Modelos No Supervisados

```
kmeans = KMeans(n_clusters=5).fit(data_3)
centroids = kmeans.cluster_centers_
print(centroids)
```

Fig. 16. Modelo K-Means

```
from sklearn.cluster import DBSCAN
from sklearn.metrics import silhouette_score

# Asumiendo que las columnas 20, 21 y 22 son tus características de interés
X = data_3.iloc[:, [1, 2, 3]]

# Creando el modelo DBSCAN
dbscan = DBSCAN(eps= 0.009, min_samples=9)

# Ajustando el modelo
dbscan.fit(X)

# Calculando el coeficiente de silueta
silhouette = silhouette_score(X, dbscan.labels_)

print('Coeficiente de silueta para DBSCAN con eps=0.009 y min_samples=9:', silhouette)
```

Fig. 17. Modelo DBSCAN

Curvas de aprendizaje

Fig. 18. Curva de Aprendizaje Modelo de Regresión

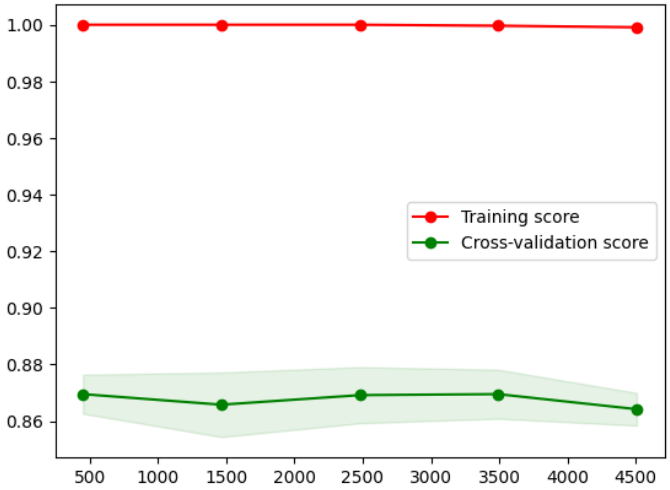

Fig. 19. Curva de Aprendizaje Modelo Random Forest

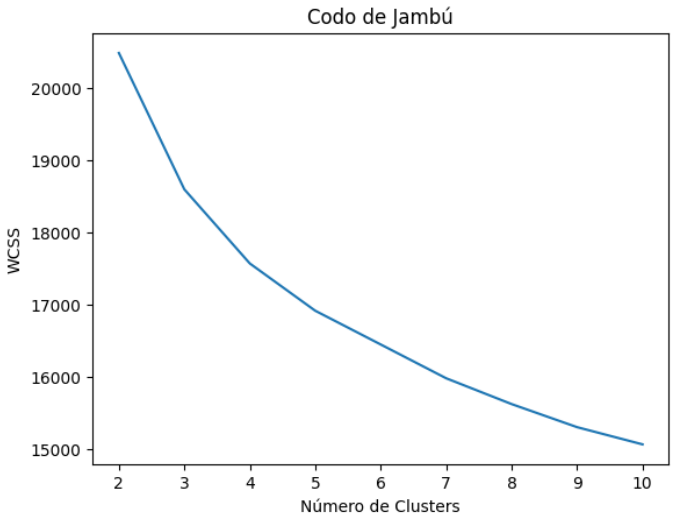

Fig. 20. Codo de Jambú Modelo K-Means

Resultados y Métricas Modelos No Supervisados

Mejores hiperparámetros (K-Means)

```
Los mejores hiperparámetros para PCA + KMeans son: {'kmeans__n_clusters': 4, 'pca__n_components': 2}
```

Fig. 21. Evaluación del Modelo K-Means

```
Puntuación de silueta para KMeans: 0.7215884794438193
Puntuación de silueta para DBSCAN: 0.5730777987189565
```

Fig. 22. Puntuaciones de Silueta para ambos modelos no supervisados

Evaluación diagnóstica

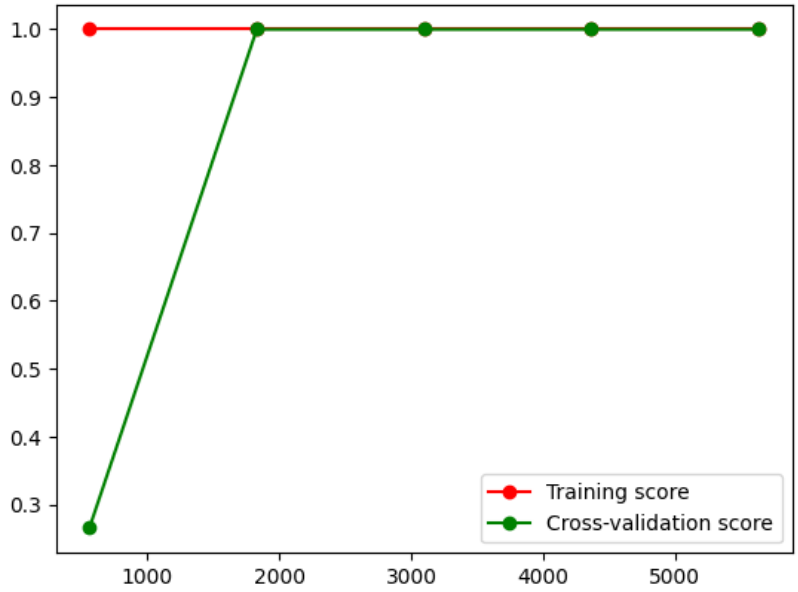

Fig. 23. Curvas de aprendizaje, que muestran el rendimiento del modelo en los conjuntos de entrenamiento y prueba

- La línea roja representa el "**Training score**", es decir, el rendimiento del modelo en el conjunto de entrenamiento. Si esta línea es alta, significa que el modelo está aprendiendo bien de los datos de entrenamiento.

- La línea verde representa el "**Cross-validation score**", es decir, el rendimiento del modelo en un conjunto de validación que no se utiliza para el entrenamiento. Esta línea da una buena idea de cómo el modelo generalizará a nuevos datos.

- En este caso, ambas líneas están en el valor máximo de 1.0, lo que indica que el modelo tiene un rendimiento excelente tanto en el conjunto de entrenamiento como en el de validación. Esto podría ser una señal de que el modelo está sobre ajustando los datos, especialmente si se tiene un conjunto de datos pequeño.

Recomendaciones

Observando los resultados obtenidos, se puede deducir que el modelo está teniendo un rendimiento excelente en los datos de entrenamiento y validación. Aunque esto puede parecer ideal, también podría ser una señal de que el modelo está sobre ajustando los datos. El sobreajuste ocurre cuando un modelo aprende tan bien los datos de entrenamiento que no generaliza bien a nuevos datos. Aquí se mencionan algunas recomendaciones para mejorar el rendimiento del modelo:

1. **Reducir la complejidad del modelo**: Si el modelo está sobre ajustado, se puede intentar reducir la complejidad del modelo. Esto podría implicar eliminar algunas características, utilizar un modelo más simple o aumentar la regularización.

2. **Recopilar más datos**: Si es posible, recopilar más datos puede ayudar a mejorar el rendimiento del modelo. Los modelos de machine learning suelen obtener mejor rendimiento con mayor cantidad de datos.

3. **Probar diferentes algoritmos**: Cada algoritmo de machine learning tiene sus propias fortalezas y debilidades, y algunos algoritmos pueden funcionar mejor en ciertos tipos de datos o problemas que otros.

Evaluación Sobre el Despliegue del Modelo en Producción

Desplegar un modelo de machine learning en producción puede ser un desafío, pero también es un paso crucial para obtener valor de los modelos. Aquí se enumerarán algunas consideraciones que se podrían tener en cuenta:

1. **Establecer un nivel de rendimiento mínimo**: Antes de desplegar un modelo en producción, se debe establecer un nivel de rendimiento mínimo que el modelo debe alcanzar. Este nivel dependerá de las necesidades específicas de la aplicación. Por ejemplo, si se está intentado conocer si una cantidad determinada de clientes abandonarán o no una compañía de telecomunicaciones según información demográfica y personal, se podría decidir que el

modelo debe ser capaz de predecir correctamente la tasa de abandono de los clientes al menos el 70% de las veces.

2. **Despliegue del modelo**: El proceso de despliegue del modelo puede variar dependiendo de la infraestructura. Algunas empresas utilizan servidores en la nube para alojar sus modelos, mientras que otras pueden tener sus propios servidores locales. Independientemente de la infraestructura que se utilice, es importante asegurarse de que el modelo esté alojado en un entorno que pueda manejar la carga de trabajo prevista.

3. **Monitoreo del rendimiento en producción**: Una vez que el modelo está en producción, es importante seguir monitorizando su rendimiento para asegurarse de que sigue funcionando como se espera. se puede hacer esto recogiendo regularmente datos de retroalimentación y utilizándolos para evaluar el rendimiento del modelo. Si el rendimiento del modelo comienza a disminuir, puede ser necesario reentrenarlo con nuevos datos.

4. **Actualización del modelo**: Los modelos de machine learning no son estáticos. A medida que se recogen más datos, es probable que se necesite reentrenar y actualizar el modelo para asegurarse de que sigue siendo relevante. Esto puede implicar ajustar los hiperparámetros del modelo, añadir nuevas características o incluso cambiar a un tipo de modelo completamente diferente.

Conclusiones

- Si nos ubicamos en el primer Notebook (01 - exploración de datos.ipynb) en su apartado de "Gráfica Preliminar del Negocio (Exploración de variables)" observamos que la mayor tasa de abandono con respecto al tipo de contrato adquirido se dio durante el primer mes, en dicho tiempo, los clientes permanecieron en un 57%, pero dejaron la compañía en un 43%, lo que indica que la empresa durante este primer mes deberá lanzar estrategias de mercadeo donde realice promociones u ofertas que inciten a los clientes a permanecer en la empresa, y así disminuir dicha tasa de abandono durante este lapso de tiempo.
- En dicho apartado también nos indica, que el servicio de internet preferido por los usuarios es la fibra óptica (44%), luego DSL (34%) y el 22% de los usuarios no tiene ninguno de los dos; básicamente el 78% de los clientes cuentan con servicio de internet.
- Con la generación de un primer modelo de predicción (Regresión logística) establecido en el segundo Notebook (02 - preprocesado. ipynb) nos dimos cuenta que ningún cliente de alto valor (CLTV) abandonaría la compañía.
- Según el Modelo de Random Forest, establecido al final del tercer Notebook (03 - modelos supervisados. Ipynb), el 75% de los hombres permanecerán en la compañía y el 25% la abandonarán. Con respecto al género femenino alude a que el 24% de las mujeres podrían dejar a la empresa y el 76% de estas, se quedaran.
- La curva de aprendizaje del modelo de Regresión Logística (Fig.18) nos dice que a medida que el tamaño del conjunto de entrenamiento aumenta, la precisión del modelo en el conjunto de entrenamiento disminuye ligeramente, lo cual es normal ya que es más difícil ajustar a un mayor número de datos. Sin embargo, la precisión en el conjunto de validación cruzada aumenta, lo que indica que el modelo está aprendiendo de los datos adicionales y generalizando mejor a datos no vistos. La gráfica también nos indica que no sería adecuado entrenar al modelo después de 2500 conjuntos de datos, puesto que posterior a esta cifra el modelo podría no beneficiarse significativamente de más datos de entrenamiento. En este

punto, podrías considerar otras formas de mejorar el rendimiento del modelo, como ajustar los parámetros del modelo, probar un modelo diferente, o recoger más características de los datos.

- La curva de aprendizaje del modelo Random Forest (Fig.19) nos demuestra que a medida que el tamaño del conjunto de entrenamiento aumenta, la precisión del modelo en el conjunto de entrenamiento se mantiene constante (cerca de 1.0), lo cual es típico para los modelos de Random Forest ya que son propensos a sobreajustar a los datos de entrenamiento. Por otro lado, la precisión en el conjunto de validación cruzada aumenta inicialmente y luego parece estabilizarse. Esto sugiere que el modelo está aprendiendo de los datos adicionales al principio, pero después de cierto punto, más datos de entrenamiento no mejoran significativamente la precisión en el conjunto de validación cruzada. Lo cual, podríamos considerar otra manera de mejorar el rendimiento del modelo.
- Los modelos no supervisados como K-Means y DBSCAN son algoritmos establecidos para observar si los datos están bien concentrados, cohesionados o agrupados, a estos les interesa es demostrar en donde se encuentra la mayor concentración de los datos y como dividirlos en Clústeres o grupos. Es por ello, que la técnica del codo de Jambú (Fig.20) del modelo K-Means sugiere el número óptimo de grupos. En este caso, parece que el codo está alrededor de 4 o 5, lo que sugiere que 4 o 5 podría ser un buen número de agrupación para los datos.
- El valor de silueta es una métrica que nos dice lo bien como se agrupan los datos. Es por ello, que el cuarto Notebook (04 - modelos_no_supervisados. ipynb) en el apartado de "Evaluando Modelos de Agrupamiento" se ve reflejado que la puntuación de silueta para K-Means es 0.72, lo que indica que los grupos formados por K-Means son densos y bien separados (Fig.8.0). La puntuación de silueta para DBSCAN es 0.57, lo que indica que los grupos formados por DBSCAN son razonablemente densos y bien separados (Fig.9.0)., pero no tanto como los grupos formados por K-Means. Esto sugiere que, para este conjunto de datos y estos parámetros, K-Means ha hecho un mejor trabajo al agrupar los datos que DBSCAN.

Referencia

IBM. (23 de Mayo de 2020). *Kaggle*. Obtenido de Kaggle: https://www.kaggle.com/datasets/yeanzc/telco-customer-churn-ibm-dataset