Clasificación de clientes de un banco

Aprendizaje supervisado con Árboles de Decisión y KNN

Yuu R. Akachi Tanaka
A01351969
Tec de Monterrey Campus Mty
Monterrey NL Mexico
A01351969@tec.mx

Donnet Hernández Franco
A01352049
Tec de Monterrey Campus Mty
Monterrey NL Mexico
A01352049@tec.mx

Pablo Monzon Terrazas
A01562619
Tec de Monterrey Campus Mty
Monterrey NL Mexico
A01562619@tec.mx

ABSTRACT

Varios bancos de todo el mundo necesitan saber si sus clientes seguirán formando parte de su banco o por el contrario, cierran la cuenta. Esto con el objetivo de poder tomar decisiones financieras sumamente importantes para el desarrollo de la empresa.

El presente artículo tiene como objetivo comparar dos modelos de clasificación: árboles de decisión y K-Nearest Neighbors, con el fin de determinar cuál de ellos ofrece una mejor solución para la clasificación de clientes de un banco en relación a su permanencia. Para ello, se trabajó con una base de datos de clientes de un banco, en la cual se identificaron las variables más relevantes para la permanencia de los clientes.

Se crearon dos modelos por defecto para cada método de clasificación y se midió su precisión en términos de la tasa de aciertos, la matriz de confusión y curva ROC-AUC. Posteriormente, se ajustaron los hiperparámetros para cada modelo con el fin de mejorar la precisión.

Los resultados indican que el modelo de árboles de decisión y K-Nearest Neighbors son capaces de clasificar de manera medianamente precisa a los clientes según su permanencia en el banco, aunque con una ligera ventaja para el árbol de decisión mejorada. Esto sugiere que el modelo de árboles de decisión ajustada es una opción más efectiva que K-Nearest Neighbors para la clasificación de clientes de bancos según sus métricas de desempeño. Es decir, el árbol de decisión es el clasificador más conveniente para nuestra base de datos.

CCS CONCEPTS

• Python • Pandas • Scikit-learn • DecisionTreeClassifier • KNeighborsClassifier

PALABRAS CLAVE

Árboles de decisión, Clasificación clientes de banco, Aprendizaje supervisado, Modelo de Clasificación, K-Nearest Neighbors

1 Introducción

Dentro de la industria bancaria es un problema crucial la predicción de la rotación de clientes. En este artículo se abordará la clasificación de una base de datos de un banco con el objetivo de predecir si un cliente continuará o no formando parte del banco. Esto permitirá a los bancos tomar decisiones de vital importancia sobre cómo retener a sus clientes y mejorar su satisfacción.

Para lograr esto, se estará utilizando un proceso de aprendizaje supervisado el cual nos estará clasificando cada cliente del banco, si cerraron su cuenta o permanecen en el banco, por lo que será una clasificación binaria. En esta investigación se estará empleando árboles de decisión el cual es una técnica ampliamente utilizada para algoritmos de aprendizaje automático. Estas se representan en forma de árbol, en donde cada nodo del árbol es una condición y las ramas representan las posibles decisiones. De igual forma, se estará comparando los resultados del modelo con otro proceso de aprendizaje supervisado, K-Nearest Neighbors, el cual funciona buscando los *k* puntos de datos más cercanos.

El dataset que se estará manejando es una base de datos que se extrajo desde la página web de Kaggle, el cual podemos encontrar en la siguiente liga: https://www.kaggle.com/datasets/shrutimechlearn/churn-modelling. Esta cuenta con 14 variables y 10,000 registros. La base de datos contiene algunas variables que no serán de nuestro interés y otros donde se debe realizar una transformación de datos categóricos a numéricos para poder realizar la clasificación, posteriormente podremos apreciar los pasos que se siguieron.

2 Conceptos previos

Para la realización de un modelo de clasificación de la permanencia o no de los usuarios dentro del banco, se necesitan conocimientos previos de algoritmos de Machine Learning, específicamente utilizaremos un algoritmo de Árbol de decisión. Este algoritmo de clasificación se basa en la utilización de la entropía, observar *Ecuación (1)*, la cual es el nivel de incertidumbre dentro de un conjunto de datos. La entropía nos permite obtener la ganancia de información una vez que se introduzca un atributo para la correcta clasificación de nuestro dataset, observar *Ecuación (2)*. Una vez obtenida la ganancia de información por medio de los atributos, se pueden detectar aquellos que ejercen un papel importante para la clasificación de los mismos y creando así las ramas del árbol de decisión como de igual forma los siguientes nodos con distintos atributos que aporten información a la clasificación, repitiendo así los mismos pasos. [1]

$$Entropy(S) = -\sum_{c \in C} p(c) log_2 p(c) \qquad (1)$$

$$Gain(S, a) = Entropy(S) - \sum_{v \in values(a)} \frac{|S_v|}{|S|} Entropy\left(S_v\right) \quad (2$$

El algoritmo de K-Nearest Neighbors utiliza la proximidad y distancia para realizar las clasificaciones sobre la agrupación de un punto de datos individual, es decir identifica los “vecinos” más cercanos de un punto a clasificar. Para realizar esto, utiliza métricas de distancia, principalmente la Distancia Minkowski que es la forma generalizada para medir distancias, véase *Ecuación (3)*. Primeramente, calcula la distancia entre el punto a clasificar con los demás puntos, ordena de menor a mayor distancia, selecciona los primeros k distancias y clasifica el nuevo punto a la clase con más cantidad. [3]

$$Minkowski\ Distance = \left(\sum_{i=1}^{n} \left| x_i - y_i \right|^p \right)^{1/p} \qquad (3)$$

De igual forma se necesita de conocimiento en el lenguaje de programación Python, puesto que dentro de este lenguaje existe una librería destinada a algoritmos de Machine Learning, scikit-learn, lo que nos permitirá realizar proceso de procesamiento de los datos, así como la modelación del algoritmo. De igual forma se utilizarán otras librerías como lo son pandas para la importación del dataset, matplotlib para la visualización de los datos y numpy para la vectorización de los datos.

3 Metodología

Para la realización de este proyecto de ciencia de datos se utilizó como base la metodología CRISP-DM, la cual permite una efectiva organización dentro del equipo de ciencia de datos para la división de tareas con el propósito de obtener un modelo de calidad y eficiente.

Figura 1: **Metodologia CRISP-DM**

Como primer paso se tiene el entendimiento del negocio, el cual nos permite detectar con qué propósito se realiza este proyecto de ciencia de datos. En nuestro caso se busca generar valor dentro de la administración del banco, el cual estamos utilizando el dataset para la toma de decisiones de forma que más usuarios permanezcan en el banco.

Una vez detectado el objetivo con el que se realiza el proyecto de ciencia de datos, se hace una exploración de los datos para verificar la calidad de los mismos, se realiza la estadística descriptiva para detectar la distribución de los datos, así como su comportamiento. Además se realiza la descripción de las variables, así como la visualización de los datos para la identificación de la distribución de las variables.

Para la preparación de los datos se realizó una normalización con el objetivo de amortiguar la diferencia de los rangos en las distintas variables, también se realizará una transformación de las variables categóricas a variables numéricas discretas, de forma que los datos puedan ser asimilados por el algoritmo de Machine Learning. Para este paso el uso de scikit-learn fue de gran importancia gracias a la función de normalización, así como el uso de la función replace de pandas.

Para la modelación, una vez que se tienen los datos preparados para la introducción de los mismos al modelo, se hace la división del conjunto de datos para el entrenamiento del modelo con un subconjunto llamado train-test. Este subconjunto permitirá la realización del árbol

de decisión y KNN que permitirá a la industria bancaria la predicción de la retención de los clientes.

En la evaluación del modelo se utilizan métricas de desempeño que se encuentra en la librería de scikit-learn, métricas como lo son precision score, f1 score, recall, matriz de confusión, valor AUC, así como la curva ROC. Y de ser necesario se calibra y mejora el modelo.

Finalmente, se despliega el modelo para poder ser utilizado por los bancos.

4 Resultados

4.1 Comprensión de los datos

Primeramente, se exploró a detalle todas las variables presentes en el dataset. A continuación se presentan las variables con su respectivo significado:

- RowNumber: Variable categórica que enumera los registros, toma valores de 1 a 10000.
- CustomerId: Variable categórica que indica la matrícula de cada cliente.
- Surname: Variable categórica que indica el apellido del cliente.
- CreditScore: Variable numérica que indica puntaje de crédito que tiene el cliente
- Geography: Variable categórica que muestra el país de residencia del cliente.
- Gender: Variable categórica que muestra si el cliente es hombre o mujer.
- Age: Variable numérica que muestra la edad del cliente.
- Tenure: Variable numérica que refleja el número de años que el cliente ha estado con el banco.
- Balance: Variable numérica que muestra el balance bancario que tiene el cliente
- NumOfProducts: Variable numérica que describe la cantidad de productos del banco que el cliente está utilizando
- HasCrCard: Variable booleana que muestra si el cliente tiene tarjeta de crédito o no.
- IsActiveMember: Variable booleana que representa si el cliente es un miembro activo o no.
- EstimatedSalary: Variable numérica que estima la cantidad de salario que tiene el cliente.
- Exited: Variable booleana que muestra si el cliente cerró o no su cuenta de banco.

Nuestra variable objetivo será *Exited* para poder clasificar si un cliente permanece o no en el banco. Las variables independientes serán *CreditScore, Geography, Gender, Age, Tenure, Balance, NumOfProducts, HasCrCard, IsActiveMember* y *Estimated Salary.* No se estarán utilizando las variables *RowNumber, CustomerId* y *Surname* ya que no aportan ninguna información valiosa para la clasificación.

Utilizando la función .info pudimos conocer que todas las variables tienen 10,000 valores no nulos y el tipo de datos que contiene cada variable, observar *Figura 2*.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 14 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   RowNumber        10000 non-null  int64
 1   CustomerId       10000 non-null  int64
 2   Surname          10000 non-null  object
 3   CreditScore      10000 non-null  int64
 4   Geography        10000 non-null  object
 5   Gender           10000 non-null  object
 6   Age              10000 non-null  int64
 7   Tenure           10000 non-null  int64
 8   Balance          10000 non-null  float64
 9   NumOfProducts    10000 non-null  int64
 10  HasCrCard        10000 non-null  int64
 11  IsActiveMember   10000 non-null  int64
 12  EstimatedSalary  10000 non-null  float64
 13  Exited           10000 non-null  int64
dtypes: float64(2), int64(9), object(3)
memory usage: 1.1+ MB
```

Figura 2: **Output de función .info()**

Asimismo, hacemos uso de la función .isnull().sum() para ver que, efectivamente, nuestro dataset no contiene valores nulos, observar *Figura 3.*

```
RowNumber          0
CustomerId         0
Surname            0
CreditScore        0
Geography          0
Gender             0
Age                0
Tenure             0
Balance            0
NumOfProducts      0
HasCrCard          0
IsActiveMember     0
EstimatedSalary    0
Exited             0
dtype: int64
```

Figura 3: **Output de función .isnull().sum()**

Posteriormente, se realizó la estadística descriptiva utilizando la función .describe() para las variables numéricas, observar *Figura 4*.

	RowNumber	CustomerId	CreditScore	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
count	10000.00000	1.000000e+04	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.00000	10000.000000	10000.000000	10000.000000
mean	5000.50000	1.569094e+07	650.528800	38.921800	5.012800	76485.889288	1.530200	0.70550	0.515100	100090.239881	0.203700
std	2886.89568	7.193619e+04	96.653299	10.487806	2.892174	62397.405202	0.581654	0.45584	0.499797	57510.492818	0.402769
min	1.00000	1.556570e+07	350.000000	18.000000	0.000000	0.000000	1.000000	0.00000	0.000000	11.580000	0.000000
25%	2500.75000	1.562853e+07	584.000000	32.000000	3.000000	0.000000	1.000000	0.00000	0.000000	51002.110000	0.000000
50%	5000.50000	1.569074e+07	652.000000	37.000000	5.000000	97198.540000	1.000000	1.00000	1.000000	100193.915000	0.000000
75%	7500.25000	1.575323e+07	718.000000	44.000000	7.000000	127644.240000	2.000000	1.00000	1.000000	149388.247500	0.000000
max	10000.00000	1.581569e+07	850.000000	92.000000	10.000000	250898.090000	4.000000	1.00000	1.000000	199992.480000	1.000000

Figura 4: **Output de función .describe()**

En este output podemos observar la media, desviación estándar, el mínimo y máximo y los percentiles de cada variable numérica.

Seguidamente, observamos la matriz de correlación de las variables haciendo uso de la función .corr(), ver *Figura 5.*

	CreditScore	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
CreditScore	1.000000	-0.003965	0.000842	0.006268	0.012238	-0.005458	0.025651	-0.001384	-0.027094
Age	-0.003965	1.000000	-0.009997	0.028308	-0.030680	-0.011721	0.085472	-0.007201	0.285323
Tenure	0.000842	-0.009997	1.000000	-0.012254	0.013444	0.022583	-0.028362	0.007784	-0.014001
Balance	0.006268	0.028308	-0.012254	1.000000	-0.304180	-0.014858	-0.010084	0.012797	0.118533
NumOfProducts	0.012238	-0.030680	0.013444	-0.304180	1.000000	0.003183	0.009612	0.014204	-0.047820
HasCrCard	-0.005458	-0.011721	0.022583	-0.014858	0.003183	1.000000	-0.011866	-0.009933	-0.007138
IsActiveMember	0.025651	0.085472	-0.028362	-0.010084	0.009612	-0.011866	1.000000	-0.011421	-0.156128
EstimatedSalary	-0.001384	-0.007201	0.007784	0.012797	0.014204	-0.009933	-0.011421	1.000000	0.012097
Exited	-0.027094	0.285323	-0.014001	0.118533	-0.047820	-0.007138	-0.156128	0.012097	1.000000

Figura 5: **Output de función .corr()**

Podemos notar que ninguna variable tiene una correlación alta con los demás. Asimismo, podemos crear una gráfica con un mapa de calor para interpretarlo de una manera más visual haciendo uso de la función .heatmap(), ver *Figura 6*, vemos de igual manera que no existen correlaciones fuertes.

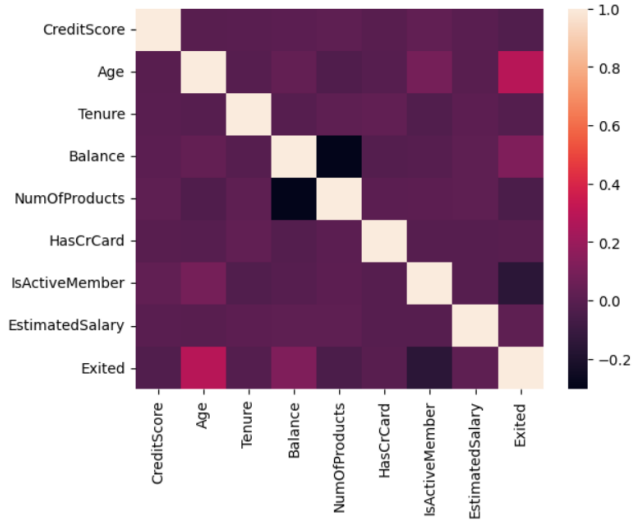

Figura 6: **Output de función .heatmap()**

A través de la visualización de histogramas podemos observar de mejor manera el comportamiento de las variables, obsérvese *Figura 7*. Vemos que algunas variables como lo es *EstimatedSalary* y *Tenure*, respetan una distribución de cierta manera uniforme. Mientras que las demás variables que no son categóricas, se puede observar que no respetan una distribución normal, ya sea gracias a que presentan asimetría o debido a que tienen una curtosis distinta a la distribución normal. Además, podemos observar que las variables *Exited, IsActiveMember* y *HasCrCard* son variables booleanas ya que únicamente tienen valores de 0 y 1.

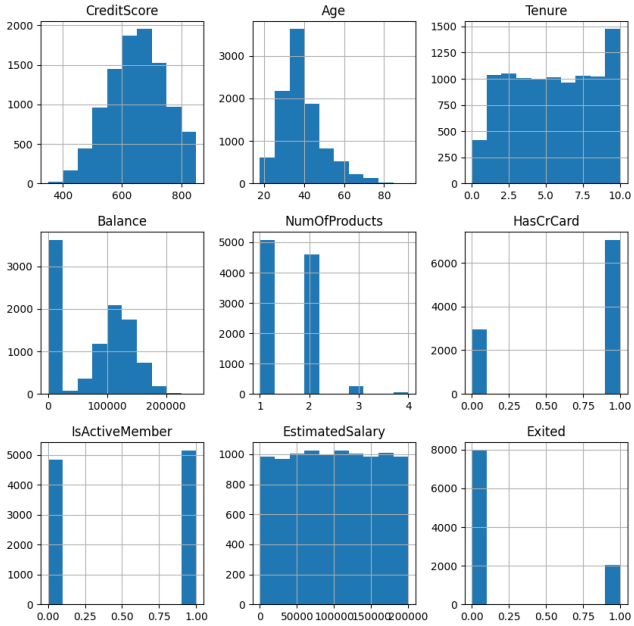

Figura 7: **Output de función .hist()**

La visualización de las variables por medio de los boxplots, ver *Figura 8,* nos permite detectar de mejor manera aquellas variables que cuentan con outliers, es decir datos que se encuentran fuera del rango del límite inferior y superior calculado por medio del rango intercuartílico. En nuestro caso no se detectan una gran cantidad de outliers a excepción de la edad, gracias a que se encuentra un gran concentración de los datos en una cierta edad.

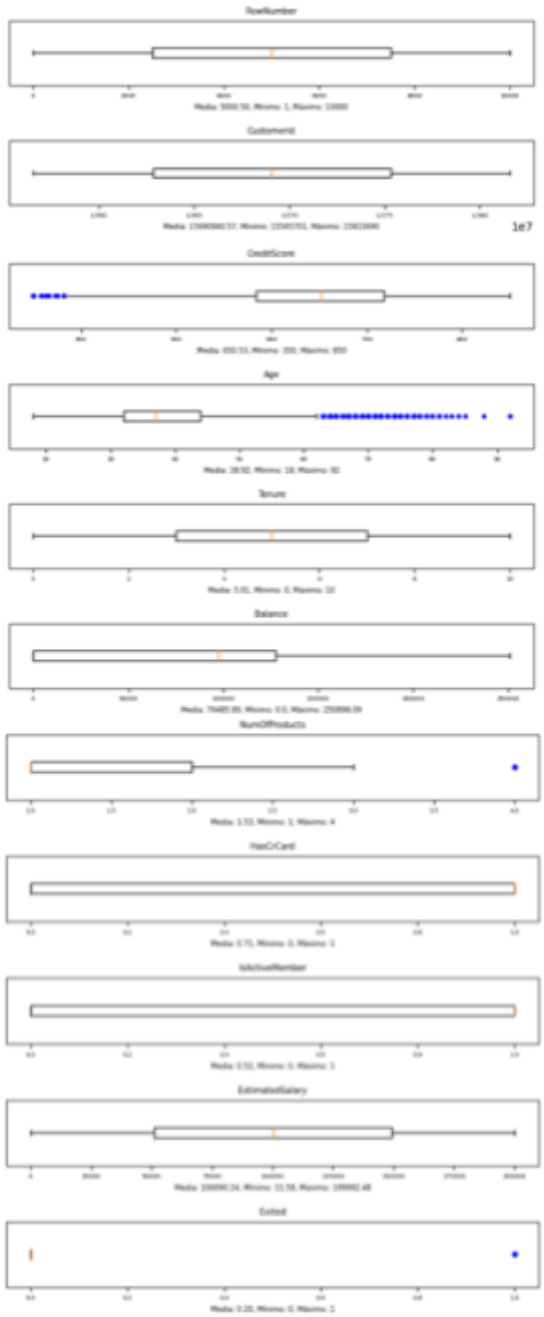

Figura 8: **Output de función .boxplot()**

En la *Figura 25* se puede observar la matriz de correlación presentada a través de gráficos de dispersión, mostrando gráficamente la relación que existe entre cada una de las variables. Observamos que ninguno logra tener una clara correlación.

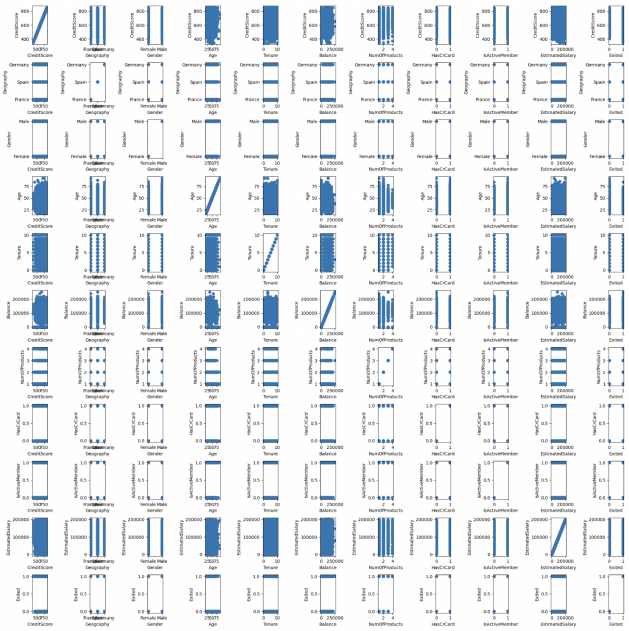

Figura 25: **Gráficos de dispersión**

4.2 Preparación y limpieza de los datos

Como primer paso para la preparación de los datos se eliminaron variables independientes que no otorgan un valor al modelo de clasificación de los datos como lo son *RowNumber, CostumerID* y *Surname*, por medio de la función .drop de Pandas, véase *Figura 9.*

```
df = df.drop(["RowNumber", "CustomerId", "Surname"], axis=1)
```

Figura 9: **Eliminación de variables que no aportan información**

Como siguiente paso para que los datos sean asimilables para el modelo de Machine Learning, se hizo la transformación de las variables categóricas a variables numéricas discretas, esto se hizo por medio de la función .replace de la librería Pandas, véase *Figura 10*. Estas variables fueron *Gender* y *Geography.*

	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
0	619	1	0	42	2	0.00	1	1	1	101348.88	1
1	608	2	0	41	1	83807.86	1	0	1	112542.58	0
2	502	1	0	42	8	159660.80	3	1	0	113931.57	1
3	699	1	0	39	1	0.00	2	0	0	93826.63	0
4	850	2	0	43	2	125510.82	1	1	1	79084.10	0
...	...	...	...	...	...	...	...	...	...	...	...
9995	771	1	1	39	5	0.00	2	1	0	96270.64	0
9996	516	1	1	35	10	57369.61	1	1	1	101699.77	0
9997	709	1	0	36	7	0.00	1	0	1	42085.58	1
9998	772	3	1	42	3	75075.31	2	1	0	92888.52	1
9999	792	1	0	28	4	130142.79	1	1	0	38190.78	0

Figura 10: **Transformación a variable numérica discreta**

En el caso de *Gender* se le asignó el valor de 0 a Female y 1 a Male. Mientras que para *Geography* se asignó el valor de 1 a France, 2 a Spain y 3 a Germany.

De igual forma se realizó la normalización de las variables numéricas continuas, de forma que se amortigüe el impacto de la diferencia de rangos entre las variables. Esto se realizó por medio de la librería de scikit-learn, véase *Figura 11*.

	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary
0	-0.326221	1	0	0.293517	-1.041760	-1.225848	1	1	1	0.021886
1	-0.440036	2	0	0.198164	-1.387538	0.117350	1	0	1	0.216534
2	-1.536794	1	0	0.293517	1.032908	1.333053	3	1	0	0.240687
3	0.501521	1	0	0.007457	-1.387538	-1.225848	2	0	0	-0.108918
4	2.063884	2	0	0.388871	-1.041760	0.785728	1	1	1	-0.365276
...	...	...	...	...	...	...	...	...	...	...
9995	1.246488	1	1	0.007457	-0.004426	-1.225848	2	1	0	-0.066419
9996	-1.391939	1	1	-0.373958	1.724464	-0.306379	1	1	1	0.027988
9997	0.604988	1	0	-0.278604	0.687130	-1.225848	1	0	1	-1.008643
9998	1.256835	3	1	0.293517	-0.695982	-0.022608	2	1	0	-0.125231
9999	1.463771	1	0	-1.041433	-0.350204	0.859965	1	1	0	-1.076370

Figura 11: **Normalización de las variables continuas**

No se consideró necesario la eliminación de outliers, gracias a la definición de las variables donde se encontraban un gran número de valores atípicos, como es el caso de *Age* y *CreditScore*. Variables que explican el comportamiento del conjunto de datos, donde en la variable *Age* hay una mayor concentración de adultos jóvenes en el banco, pero no se pueden descartar los valores atípicos que representan aquellos adultos mayores. Esto gracias a que los valores extremos de personas mayores a 90 años se contempló que si es posible que formen parte de la base de datos del banco y no sea un error de registro, ya que pueden permanecer en el banco por medio de un plan de jubilación.

De igual forma, consideramos pertinente mantener los registros de aquellos outliers en la variable *CreditScore* gracias a que existe una gran concentración de clientes con una buena salud financiera pero también es posible que existen clientes que no cuenten con una óptima salud financiera, y eliminarlos podríamos privar de información relevante para el entrenamiento del modelo.

4.3 Modelado

Para llevar a cabo el modelado del árbol de decisión y KNN, primeramente se separó el dataset en los datos de entrenamiento y de prueba. Para realizar esto, se utilizó la librería de scikit-learn, en donde se utilizó la función train_test_split. Se dividió el dataset en X_train, X_test, Y_train, Y_test, el cual el 80% del dataset se dividió para entrenar el modelo y el restante 20% se destinó para la prueba.

4.3.1 Árbol de decisión default

Una vez realizado esto, primeramente, se utilizó la función DecisionTreeClassifier de la librería scikit-learn para crear el modelo de clasificación. Primeramente, se realizó el modelo con los hiperparámetros default de python que son:

- crterion=”gini”,
- splitter=”best”
- max_depth=None
- min_samples_split=2
- min_weight_fraction_leaf=0
- max_features=None

Utilizando la función tree de sklearn graficamos el árbol de decisión, véase la *Figura 12*.

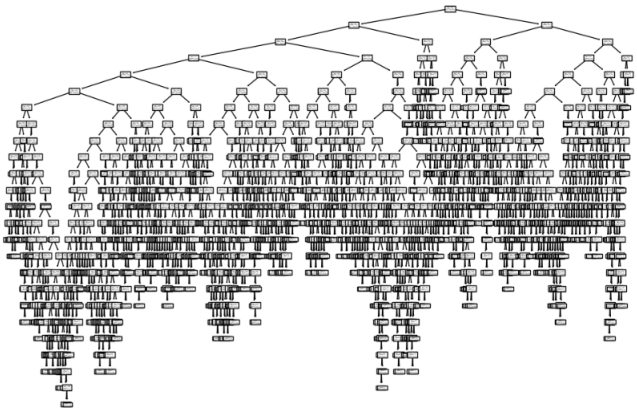

Figura 12: **Árbol de decisión con hiperparámetros default**

Una vez hecho el modelo utilizamos el array de Y_test y se compara con el Y_pred para observar cuántos clasificó bien. En este caso clasificó bien 8 de 10 registros, véase la *Figura 13*.

```
# Realizar predicciones con base en el test set
Y_pred = arbol.predict(X_test)
Y_pred[1:10]
```

```
array([0, 0, 0, 0, 0, 1, 0, 0, 0])
```

```
a = Y_test[1:10]
a.T
```

	4684	1731	4742	4521	6340	576	5202	6363	439
Exited	0	0	0	0	0	0	1	0	0

Figura 13: **Prueba Árbol de decisión con hiperparámetros default**

4.3.2 Árbol de decisión mejorado

Como se puede observar, el proceso que lleva el modelo es muy complejo, no es lo óptimo y se puede mejorar, por lo que se decidió hacer uso de la función GridSearchCV, para encontrar los hiperparámetros del modelo de Árboles de decisión para maximizar su desempeño y su eficiencia. Este es un tipo de Cross Validation que mejora el modelo.

Al aplicar esta función encontramos que los mejores hiperparámetros para el árbol de decisión para nuestro dataset son:

- criterion = “gini”
- max_depth = 6
- min_samples_split = 4

Por lo que volvemos a aplicar el modelo, pero ahora con estos hiperparámetros, dándonos la gráfica del árbol de decisión como se muestra en la *Figura 14.*

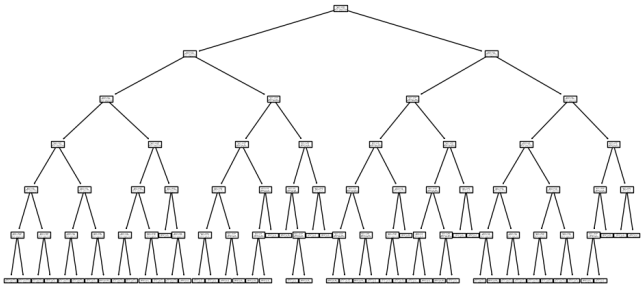

Figura 14: **Árbol de decisión con hiperparámetros mejorados**

Observamos que ahora se tienen menos ramas y nodos que el primer modelo, lo cual mejora el modelo. De la misma manera se comparó Y_test con Y_pred para tener una pequeña idea de cómo funcionó el modelo. Vemos que ahora tuvo únicamente 1 error por lo que sí mejoró nuestro modelo, véase *Figura 15.*.

```
# Realizar predicciones con base en el test set
Y2_pred = arbol2.predict(X2_test)
Y2_pred[1:10]
```

```
array([0, 0, 0, 0, 0, 0, 0, 0, 0])
```

```
y2test = Y2_test[1:10]
y2test.T
```

	4684	1731	4742	4521	6340	576	5202	6363	439
Exited	0	0	0	0	0	0	1	0	0

Figura 15: **Prueba Árbol de decisión con hiperparámetros mejorados**

4.3.3 K-Nearest Neighbors default

De igual forma, se creó un modelo con el modelo de clasificación de aprendizaje supervisado de K-Nearest Neighbors con el fin de comparar los resultados con el árbol de decisión y poder elegir el que tenga mejor rendimiento.

Para esto se realizó el modelo de clasificación por medio de la librería KNeighborsClassifier, primeramente con sus hiperparámetros default. Se realizó la predicción y se comparó con sus resultados para tener una primera vista de cómo funcionó el modelo. En donde 8 de 10 registros tuvo una clasificación correcta.

4.3.4 K-Nearest Neighbors mejorado

Posteriormente, se hizo el ajuste de sus hiperparámetros por medio de la función GridSearchCV. Dándonos un ajuste de hiperparámetros como se muestra a continuación:

- algorithm = ‘ball_tree’
- n_neighbors=9
- weights='distance'

Realizando la predicción del modelo tuvo 9 de 10 aciertos, por lo que de igual manera se tuvo una mejora en el modelo.

Observando un poco de los resultados, sabemos que los modelos con los hiperparámetros ajustados fueron los que tuvieron mejores resultados. Sin embargo, aún no podemos definir si el árbol de decisión o KNN fue el que tuvo el mejor rendimiento, por lo que se decidirá utilizando las métricas de desempeño.

4.4 Evaluación

Para la evaluación del modelo se utilizaron distintas funciones proporcionadas por la librería scikit-learn, las cuales nos permiten obtener el precision score, f1 score, recall, matriz de confusión, valor AUC, así como la curva ROC del modelo generado.

4.4.1 Árbol de decisión default

En nuestro primer modelo podemos observar que se obtienen valores muy bajos en las métricas de evaluación, véase *Figura 16*.

	Árbol de Decisión
Precisión	44.700461
Exactitud	78.050000
Sensibilidad	49.363868
F1	46.916566
Curva ROC - AUC	0.672146

Figura 16: **Métricas de rendimiento de árbol con hiperparámetros default**

Asimismo, a través de la matriz de confusión, véase *Figura 17,* nos permitió conocer los valores verdaderos positivos, falsos positivos, verdaderos negativos y falsos negativos del modelo, el cual nos hace ver que no tuvimos alto desempeño.

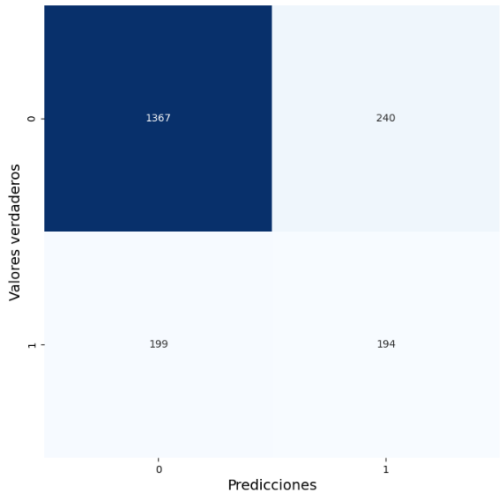

Figura 17: **Matriz de confusión de árbol de decisión con hiperparámetros default**

De igual manera, analizando la curva ROC con el valor AUC, pudimos observar que a pesar de tener un valor mayor a 0.5, en donde 1 es un modelo perfecto y 0 es un mal modelo, véase *Figura 18*, debemos ajustar el modelo para poder obtener un mejor desempeño del modelo.

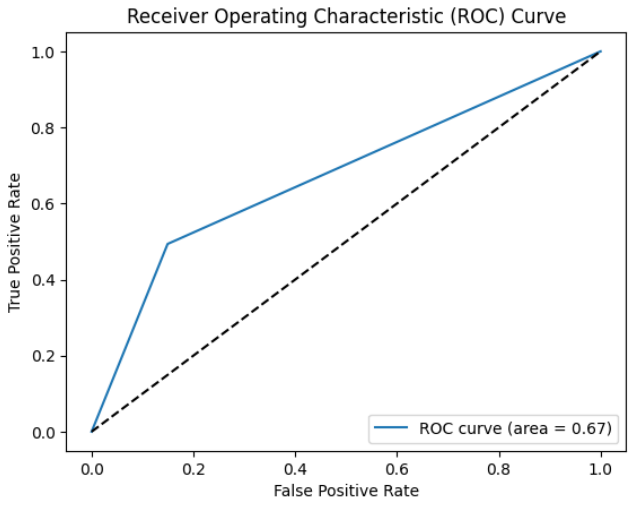

Figura 18: **Curva ROC y valor AUC de hiperparámetros default**

4.4.2 Árbol de decisión mejorado

Gracias a esto se tomó la decisión de buscar los mejores hiperparametros, como se mencionó anteriormente, lo cual mejora en gran medida las métricas de desempeño. Por lo tanto llegamos a un modelo medianamente confiable, gracias a que presenta un precision score de 73.25%, f1 score de 55.97%, recall de 45.29% y un, valor AUC de 0.70, véase de *Figura 19*.

	Árbol de Decisión Ajustado
Precisión	73.251029
Exactitud	86.000000
Sensibilidad	45.292621
F1	55.974843
Curva ROC - AUC	0.706239

Figura 19: **Métricas de rendimiento de árbol con hiperparámetros mejorados**

Para el nuevo modelo, nuestra matriz de confusión queda mejor ajustada, en donde aumentaron considerablemente los valores verdaderos positivos y verdaderos negativos, véase la *Figura 20.*

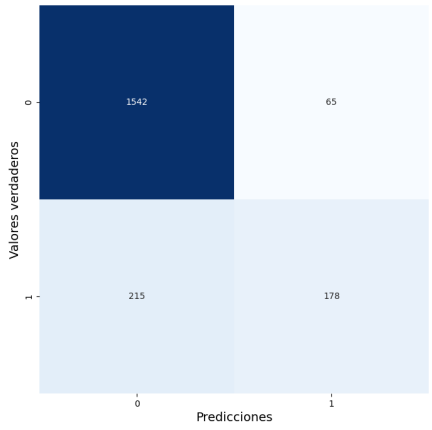

Figura 20: **Matriz de confusión de árbol de decisión con hiperparámetros mejorados**

Finalmente, vemos que la curva ROC y el valor AUC se acercan más al valor de 1 por lo que representa un mejor modelo que el primero, véase la *Figura 21.*

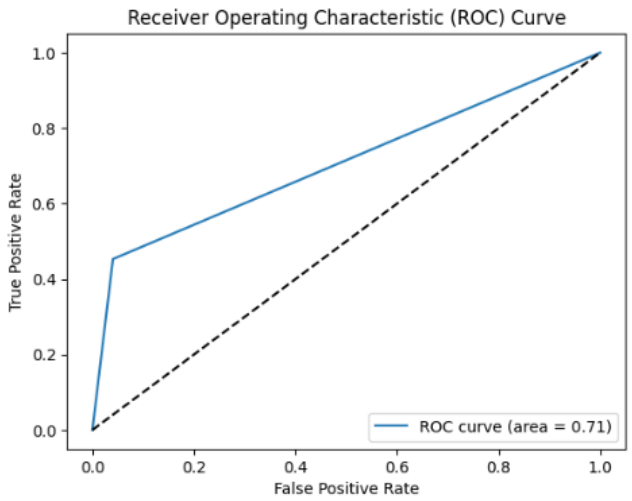

Figura 21: **Curva ROC y valor AUC de hiperparámetros mejorados**

Por lo que efectivamente, tiene un mejor rendimiento el modelo con los hiperparámetros ajustados.

4.4.3 KNN default

En el caso del algoritmo de K-Nearest Neighbors con hiperparámetros default, se pudo observar mejores resultados de rendimiento que el árbol de decisión default, véase *Figura 22.* Por lo que en principio supone un mejor algoritmo de clasificación, sin tomar en cuenta el ajuste de hiperparámetros del árbol de decisión ya que no supera el valor AUC de 0.71.

	KNN sin hiperparametros mejorados
Precisión	63.404255
Exactitud	83.500000
Sensibilidad	37.913486
F1	47.452229
Curva ROC - AUC	0.662809

Figura 22: **Métricas de rendimiento de KNN con hiperparámetros default**

Observando la matriz de confusión, véase *Figura 23*, podemos notar que efectivamente tuvo un mejor desempeño que el árbol de decisión default ya que tiene valores mayores en los verdaderos positivos y verdaderos negativos.

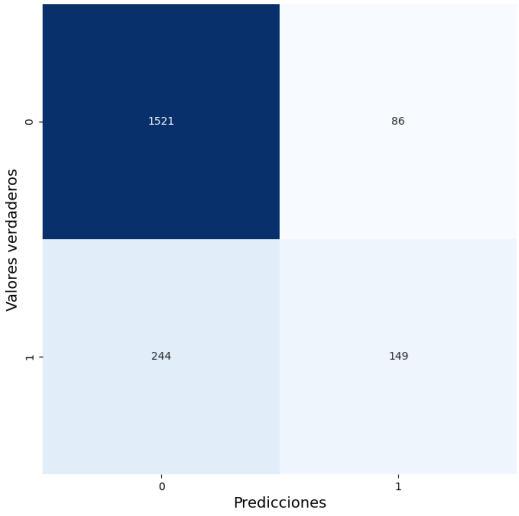

Figura 23: **Matriz de confusión de KNN con hiperparámetros default**

4.4.4 KNN mejorado

Sin embargo, después de buscar los óptimos hiperparámetros para el modelo de clasificación, se puede observar que las métricas de rendimiento del modelo son menores que el modelo de árbol de decisión mejorado. Véase *Figura 23.*

	KNN hiperparametros mejorados
Precisión	68.571429
Exactitud	84.250000
Sensibilidad	36.641221
F1	47.761194
Curva ROC - AUC	0.662671

Figura 23: **Métricas de rendimiento de KNN con hiperparámetros mejorados**

Además, observando la matriz de confusión de este modelo, véase *Figura 24,* observamos que los verdaderos positivos y los verdaderos negativos no superan al modelo de árbol de decisión mejorado.

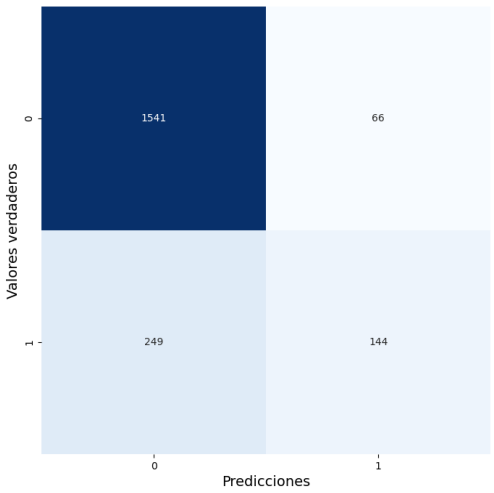

Figura 24: **Matriz de confusión de KNN con hiperparámetros mejorados**

Es por esto que consideramos como mejor modelo el del árbol de decisión mejorado. Se puede inferir que la entropía sirvió de mejor manera que el uso de distancias entre cada instancia para la clasificación de instancias en este caso en específico, fue el más adecuado para el tipo de dataset que tenemos.

5 Conclusión y reflexiones

5.1 Conclusion general

Una vez completada la metodología CRISP-DM para la elaboración de este proyecto de Ciencia de Datos, y enfocándonos principalmente en la fase de evaluación del modelo de aprendizaje supervisado. Se puede observar a través de las métricas de rendimiento, así como la curva ROC-AUC, que el modelo de árbol de decisión mejorado es el mejor método para clasificar a los clientes de un banco, en donde se predice si un cliente permanece en el banco o por el contrario cierra su cuenta. La manera más eficiente de comparar los modelos es con ayuda de la curva ROC-AUC, en donde primeramente, para ser un buen modelo debe encontrarse en la parte superior que la línea diagonal punteada, lo que nos indica que la predicción a través del modelo no se basa en el azar (línea punteada), y que se clasifica de manera correcta. Observamos que el modelo de árbol de decisión mejorada tuvo el valor AUC más alto a comparación de los otros, con un valor de 0.71.

En relación a la métrica de precisión, que indica la razón de aquellos clientes que dejaron de ser clientes del banco y que el modelo predijo correctamente, del total de las predicciones como clientes retirados del banco es superior al 50%, lo que indica que cuando se hace una predicción de los clientes que cerraron su cuenta es certero en 73 de cada 100 casos, mientras que 23 personas se predijo que dejaron de ser clientes cuando en realidad aún permanecen como clientes.

Por otra mano, la métrica de exactitud, que nos proporciona las predicciones que fueron correctas de permanencia o no en el banco del total de predicciones realizadas, fue bastante elevada con un valor de 86%. Por lo tanto, si se quiere realizar predicciones con clientes que no forman parte de la base de datos, la certeza en la clasificación, ya sea de permanencia o no de los clientes, será muy elevada respecto al total de las predicciones.

Por último la sensibilidad, métrica que nos indica la razón de las correctas predicciones de las personas que dejaron de ser clientes del banco del número total de personas que realmente dejaron de ser clientes del banco, se puede observar que tiene un valor cercano al 50% por lo cual se predice solamente la mitad de personas que realmente dejaron de ser clientes del banco [2].

Observamos que con los valores de los hiperparámetros default tuvo un mejor desempeño el modelo de K-Nearest Neighbors, no obstante, el ajuste de los hiperparámetros usando GridSearchCV favoreció al modelo de Árboles de decisión, haciendo que sea el mejor modelo para nuestra base de datos.

Por lo tanto, para realizar una clasificación con el dataset que se utilizó para esta investigación, lo más conveniente y eficiente es utilizando el aprendizaje supervisado de árboles de decisión, manejando los hiperparámetros adecuados para el problema que fueron los siguientes:

- criterion = “gini”
- max_depth = 6
- min_samples_split = 4

Así podremos crear un modelo que sea utilizable para los bancos con una exactitud de alrededor de 73.25%, superando el modelo de K-Nearest Neighbors.

5.2 Reflexiones individuales

Yuu: Después de haber realizado este trabajo de investigación, pude notar lo importante que son los modelos de Machine Learning en nuestra vida cotidiana. Los aprendizajes supervisados para la clasificación de datos son una herramienta esencial para todos nosotros. Estos nos permiten tomar las mejores decisiones para tomar acciones pertinentes para la mejora continua de las actividades, como lo es aumentar ingresos, ahorro de tiempo y esfuerzo, solucionar problemas, entre muchas otras cosas. En nuestro caso, realizar la clasificación de los clientes de banco será de gran utilidad para las empresas bancarias para que puedan retener a sus clientes, mejorar su satisfacción o incluso buscar a clientes con ciertos perfiles que puedan beneficiar a la empresa.

De igual forma, aprender sobre los métodos matemáticos que hay detrás de los aprendizajes supervisados, como lo es la entropía para los árboles de decisión y la métrica de Minkowski por parte de K-Nearest Neighbors, me hizo comprender cómo es que los clientes se distribuyen en los datos y entender qué aspectos son los más importantes en los clientes para que sean clasificados.

En los resultados de esta práctica obtuvimos que el árbol de decisión era un mejor modelo de clasificación. No obstante, debemos tomar en cuenta que siempre será importante comparar varios modelos ya que el rendimiento de los modelos depende mucho de los datos que se tienen en el dataset, ya sea por el tamaño y el tipo de datos que contengan dentro de la base de datos. Además, nosotros tuvimos una precisión de 73% en el modelo, un valor aceptable pero no perfecto ya que se aleja medianamente del 100%, sin embargo, en la vida cotidiana, habrán miles de problemas en donde realizar una clasificación sea bastante complejo, dándonos resultados poco deseados. El reto será encontrar técnicas y métodos para intentar hacer válido el modelo, como por ejemplo en este proyecto logramos mejorar el modelo utilizando un método de Cross Validation.

Para obtener un buen modelo, lo más importante es lo previo a generar el modelo. Se debe invertir bastante tiempo y dedicación en la parte de entender las variables, limpiar los datos, transformar los datos, el preprocesamiento de los datos, etc. Esto con el objetivo de que el modelo logre tener el mayor desempeño. De esto depende que nuestro modelo vaya a funcionar o no y que sea confiable o no, por lo que no todo se trata de los modelos de Machine Learning.

Hoy en día y cada vez más, los modelos de Machine Learning se están introduciendo en nuestros alrededores, teniendo un crecimiento exponencial que pueda revolucionar el mundo que conocemos. No obstante, es de vital importancia lograr ponerlos en marcha y sobre todo entenderlos, por lo que esta práctica fue bastante relevante para el desarrollo de mi conocimiento.

En cuanto a mi contribución en este trabajo, todos los miembros del equipo tuvimos una gran participación para la elaboración del artículo científico. En mi caso, pude trabajar en el código de python para la realización del modelo, participé en la descripción y la explicación de los pasos que se llevaron a cabo para este trabajo, apoyé en la interpretación de resultados y la elaboración de este documento. A mi parecer todos pudimos trabajar cómodamente, aportando todos sus conocimientos y habilidades, logrando así todo un éxito en el resultado del artículo científico.

Donnet: Como modelos de inteligencia artificial, tanto el modelo de árboles de decisión como K-Nearest Neighbors son una herramienta muy poderosa para la clasificación de datos, y complementar el uso de estos con herramientas y métodos enfocados a lograr un mejor ajuste que brinde mayor precisión al modelo (tales como la preparación de los datos, normalización, GridSearchCV, etc) puede generar resultados muy eficaces en relación al objetivo que se plantee, como en este caso, que fue un modelo de clasificación enfocado a la permanencia de los clientes de un banco.

Aunque ambos modelos produjeron resultados aceptables, la selección de un modelo óptimo fue necesaria y se basó en pruebas de precisión, exactitud y visualización de la matriz de confusión, entre otros factores. Sin embargo, es importante tener en cuenta que la elección del modelo efectivo dependerá de múltiples factores, como la naturaleza de los datos y la precisión requerida.

Considero que el estudio realizado a lo largo de este artículo es relevante en especial para el sector bancario, ya que proporciona un ejemplo aplicado de herramientas valiosas para la toma de decisiones sobre la permanencia de los clientes, lo que a su vez puede tener implicaciones importantes en la rentabilidad y la satisfacción del cliente. Además, este trabajo puede ser un punto de partida para futuras investigaciones sobre la aplicación de diferentes técnicas de clasificación en la industria bancaria y otros sectores empresariales.

En lo que respecta a mi aporte en el artículo, desempeñé un papel fundamental en el desarrollo del código en python, contribuyendo en el análisis descriptivo para el entendimiento de los datos, realizando su limpieza y preparación, y aplicando los modelos de clasificación para generar resultados precisos y detallados. Además, participé en la redacción del artículo, traduciendo los resultados obtenidos en el proceso de modelado para presentarlos de manera clara y concisa a lo largo del artículo.

Pablo: El uso de métodos matemáticos como son la entropía, el gradiente descendiente, el uso de distancias, entre otros, ha permitido la generación de algoritmos sumamente complejos que permiten realizar predicciones con base a un patrón de comportamiento de un conjunto de datos, que a simple vista no aportan ningún tipo de valor para las empresas.

En este tipo de casos se puede observar la versatilidad de las aplicaciones de las matemáticas. Versatilidad que nos permite incluso “predecir” el futuro como se trata de realizar en este caso de proyecto de ciencia de datos o simular el comportamiento de ciertos procesos humanos, como lo son las redes neuronales en modelos de aprendizaje supervisado.

El *aprendizaje* de dichos modelos a través de la detección de patrones, permite la generación de información a quien corresponda para que a partir de ella tomar decisiones y medidas en las que se vean beneficiadas las empresas. La

elaboración de este proyecto, así como las distintas actividades realizadas a lo largo de la materia, permite observar lo multidisciplinario que es la aplicación de algoritmos de Machine Learning para la mejora de procesos y servicios, en este caso se realizó en la industria bancaria, pero puede ser utilizado en la medicina, en deportes, entre otros ejemplos de bases de datos vistas antes de elegir con la que se trabajó.

El descubrimiento de esta nueva área de conocimiento ha generado una revolución en las empresas, donde anteriormente la recopilación de datos se usaban únicamente con la finalidad de mantener registros, y en la actualidad se prioriza la correcta recopilación de los datos para la generación de valor.

En nuestro caso la aplicación de Machine Learning permite proporcionar a la industria bancaria un mayor conocimiento en aquellos factores más relevantes para la retención de sus clientes, de forma que a través de estos puedan mejorar su servicio o la generación de campañas de marketing de cierto sector, de forma que cuenten con una cartera de clientes fieles a su compañía.

Mi aportación en este trabajo de Ciencia de Datos se basó en la realización de ciertas partes de las fases de la metodología CRISP-DM, específicamente en la programación en Python de la exploración y comprensión de los datos, así como la modelación y evaluación de un algoritmo de aprendizaje supervisado (KNN). De igual forma aporte en la elaboración del documento en la descripcion del proceso de comprensión y preparación de los datos, la explicación general de la metodología, entre otros puntos.

Considero que la realización de este proyecto de ciencia de datos fue muy ameno gracias a una correcta distribución de trabajo, y la comunicación continua que permitió el intercambio de ideas y conocimientos.

REFERENCIAS

[1] IBM. (2023). ¿Qué es un árbol de decisión?, de IBM. Recuperado de:https://www.ibm.com/es-es/topics/decision-trees

[2] Datos. (2021). ¿Cómo sé si mi modelo de predicción es realmente bueno?, de Datos.Gob. Recuperado de: https://datos.gob.es/es/blog/como-se-si-mi-modelo-de-prediccion-es-realmente-bueno#:~:text=Precision%3A%20esta%20m%C3%A9trica%20est%C3%A1%20determinada,%2F%20(VP%20%2B%20FP).

[3] IBM. (2023). ¿Qué es el algoritmo de k vecinos más cercanos?, de IBM. Recuperado de:https://www.ibm.com/mx-es/topics/knn