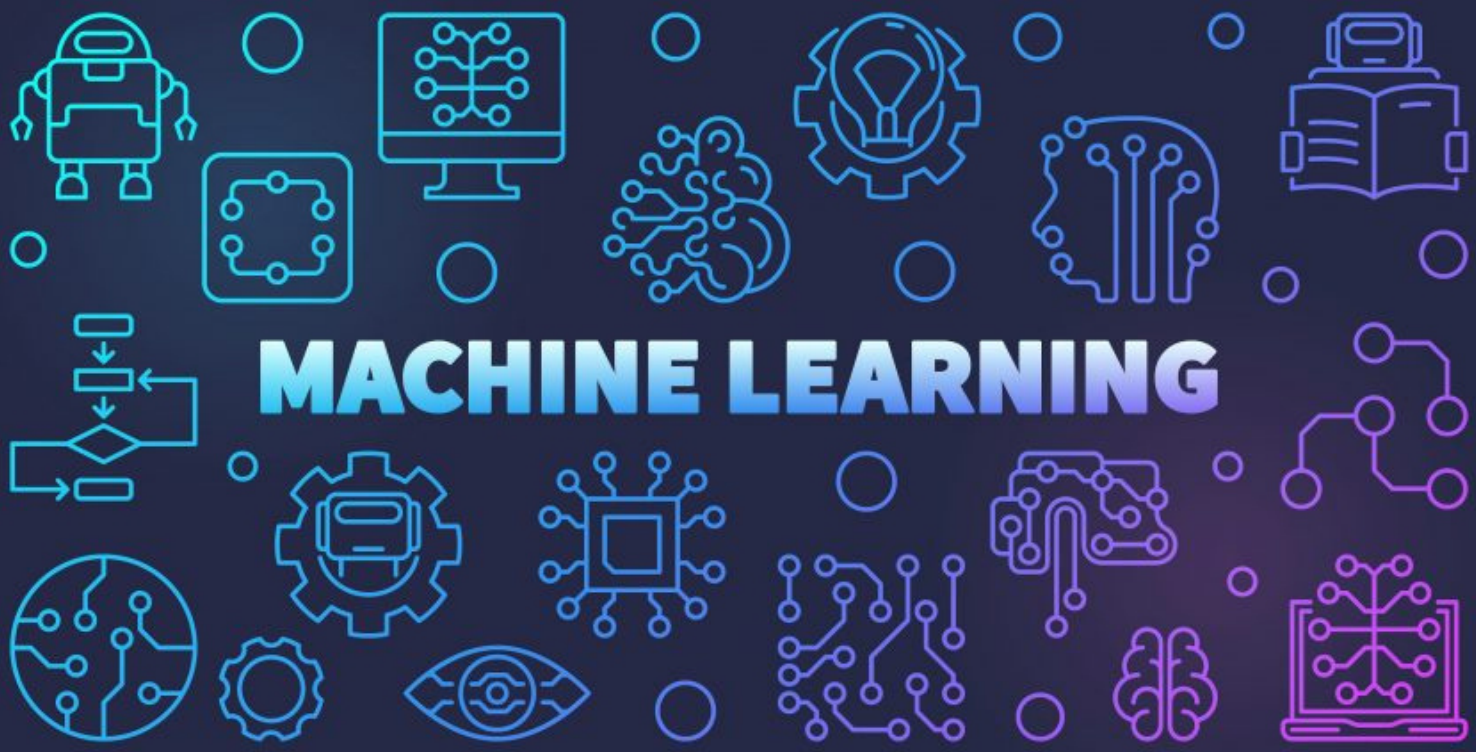



10
22

# WORKSHOP3

*por DSConsultores*

**RESUMEN**
*de los objetivos*

**MODELOS**
*de Clasificacion*

**CONCLUSION**
*del trabajo realizado*

# RESUMEN

El presente informe tiene por objetivo mostrar de manera resumida todo el trabajo realizado para arribar a un modelo de machine learning optimo que nos permita realizar la clasificacion de los clientes del banco ABC y poder predecir la tasa de abandono de los mismos.

Todo el trabajo se realizo en Python en un entorno de Jupyter Lab, aplicando los conocimientos adquiridos dentro del curso de Data Science de Digital House.

A continuacion se muestra todo el trabajo realizado y se adjunta el notebook con el codigo correspondiente.

# INDICE


1 Introducción

2 Dataset

3 Modelos No Supervisados

- 3.1 Clustering Jerarquico
- 3.2 K-Means
- 3.3 DBSCAN

4 Modelos de Clasificacion

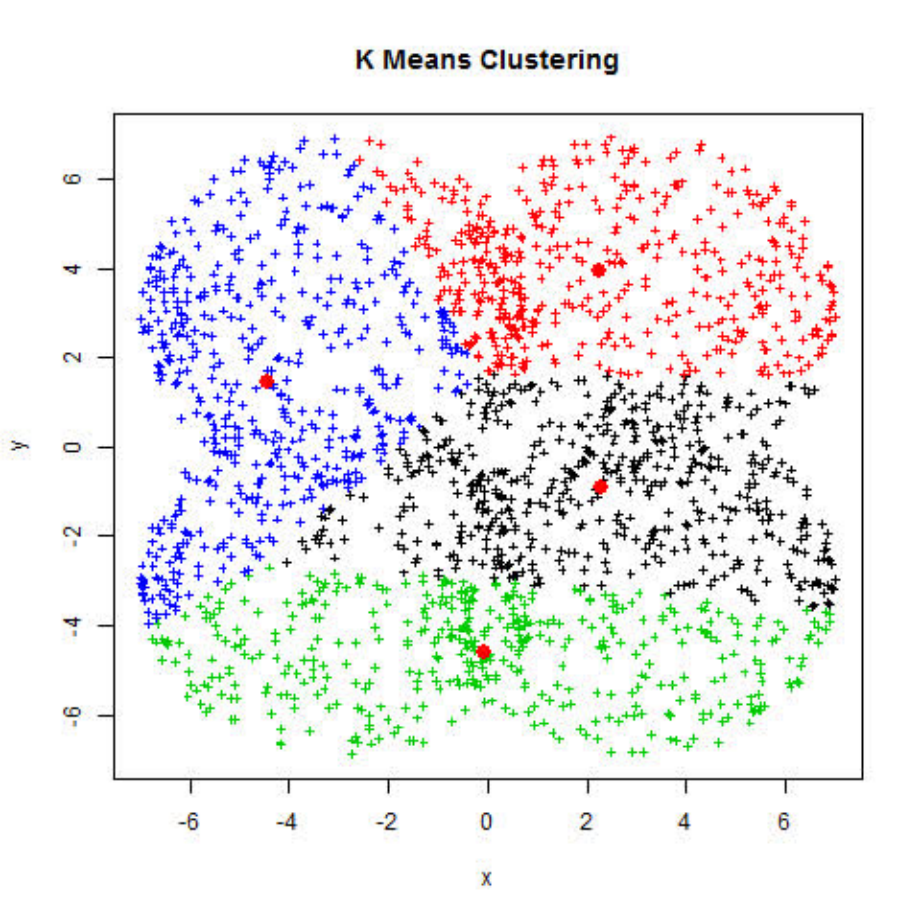
- 4.1 Naive Bayes
- 4.2 KNN
- 4.3 Regresion Logistica

5 Conclusiones

1

# INTRODUCCION

En este tercer workshop del curso de Data Science de Digital House nos enfocaremos en realizar distintos modelos de clasificacion para analizar los datos de un dataset elegido y poder arribar a conlusiones sobre los mismos, como asi tambien poder determinar ventajas y desventajas de los distintos modelos aplicados, todo esto lo haremos con los conocimientos adquiridos hasta el momento en el desarrollo del curso



2

# DATASET

*https://www.kaggle.com/datasets/gauravtopre/bank-customer-churn-dataset?resource=download*

Se realizo un analisis de todos los datasets ofrecidos y elegimos el denominado **"Bank Customer Churn"**, que se puede observar en el enlace superior.
El mismo contiene un conjunto de datos para el banco ABC Multistate con *datos bancarios de clientes para predecir la rotación de clientes* y cuenta con las siguientes columnas:

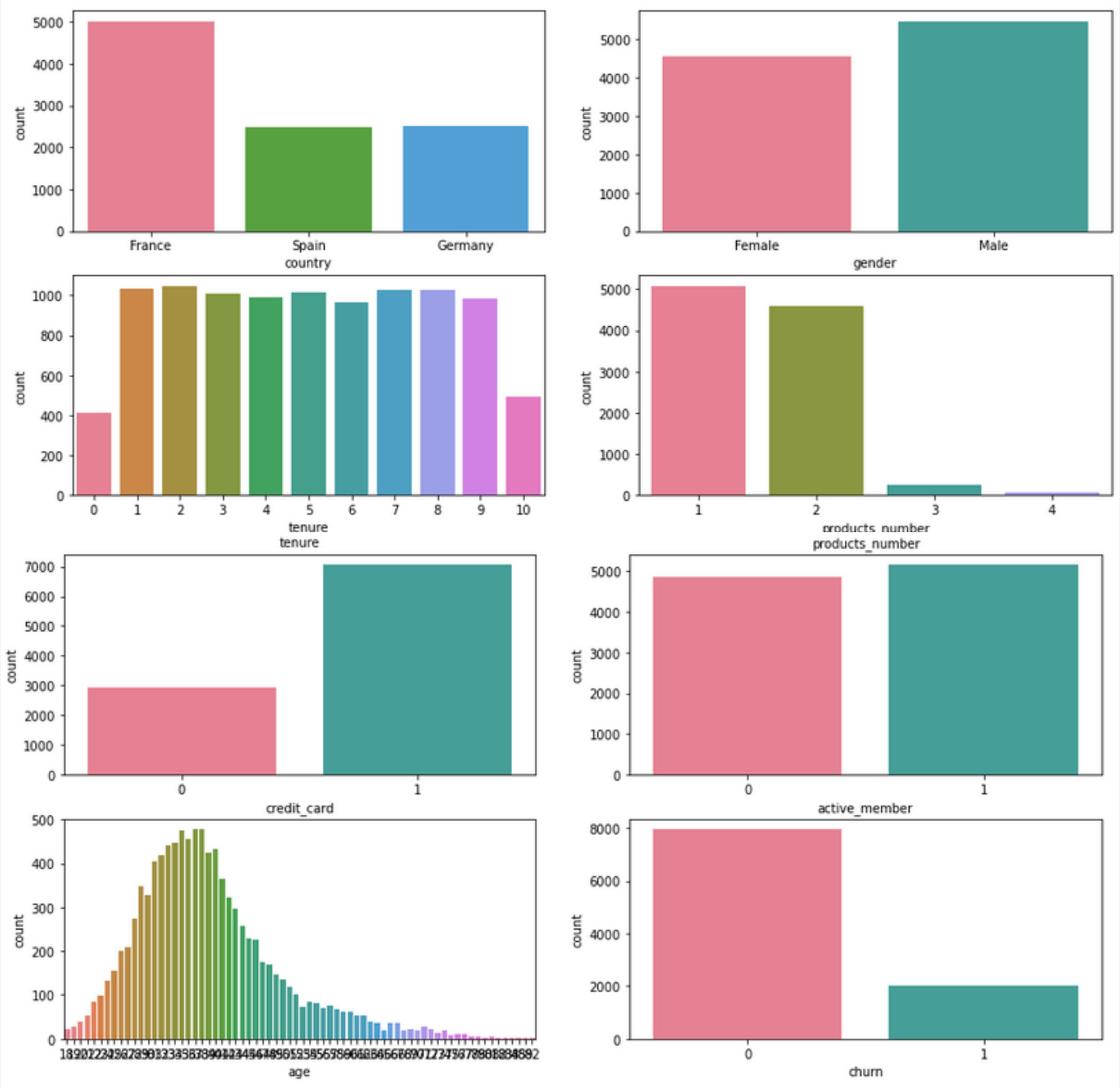
1. **customer_id**, variable sin usar.
2. **credit_score**, usada como input.
3. **country**, usada como input.
4. **gender**, usada como input.
5. **age**, usada como input.
6. **tenure**, usada como input.
7. **balance**, usada como input.
8. **products**_number, usada como input.
9. **credit_card**, usada como input.
10. **active_member**, usada como input.
11. **estimated_salary**, usada como input.
12. **churn**, usada como target. 1 si el cliente ha salido del banco durante algún tiempo o 0 si no lo ha hecho.

Realizaremos en esta etapa los pasos necesarios para: agregar las columnas que consideramos relevantes para nuestro analisis, quitar los nulos que perjudiquen al modelo y obtener el dataset lo mas prolijo posible para luego entrenar nuestro modelo.

***El OBJETIVO es predecir la tasa de abandono de clientes del Banco ABC***

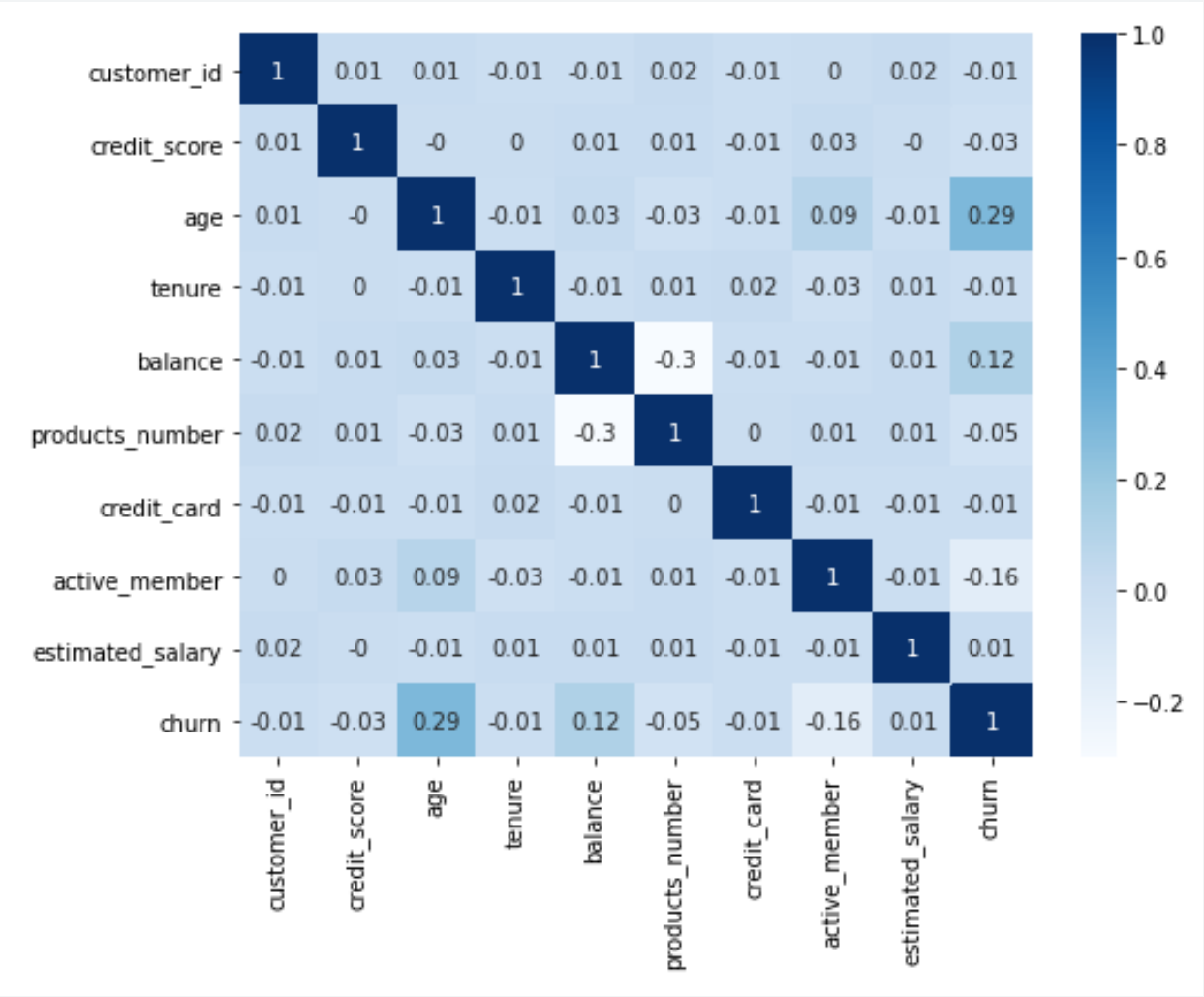
Luego de un primer analisis exploratorio podemos observar que el mismo cuenta con 10000 registros, 12 columnas y que no contiene valores nulos ni duplicados

Vemos como estan distribuidos nuestros datos



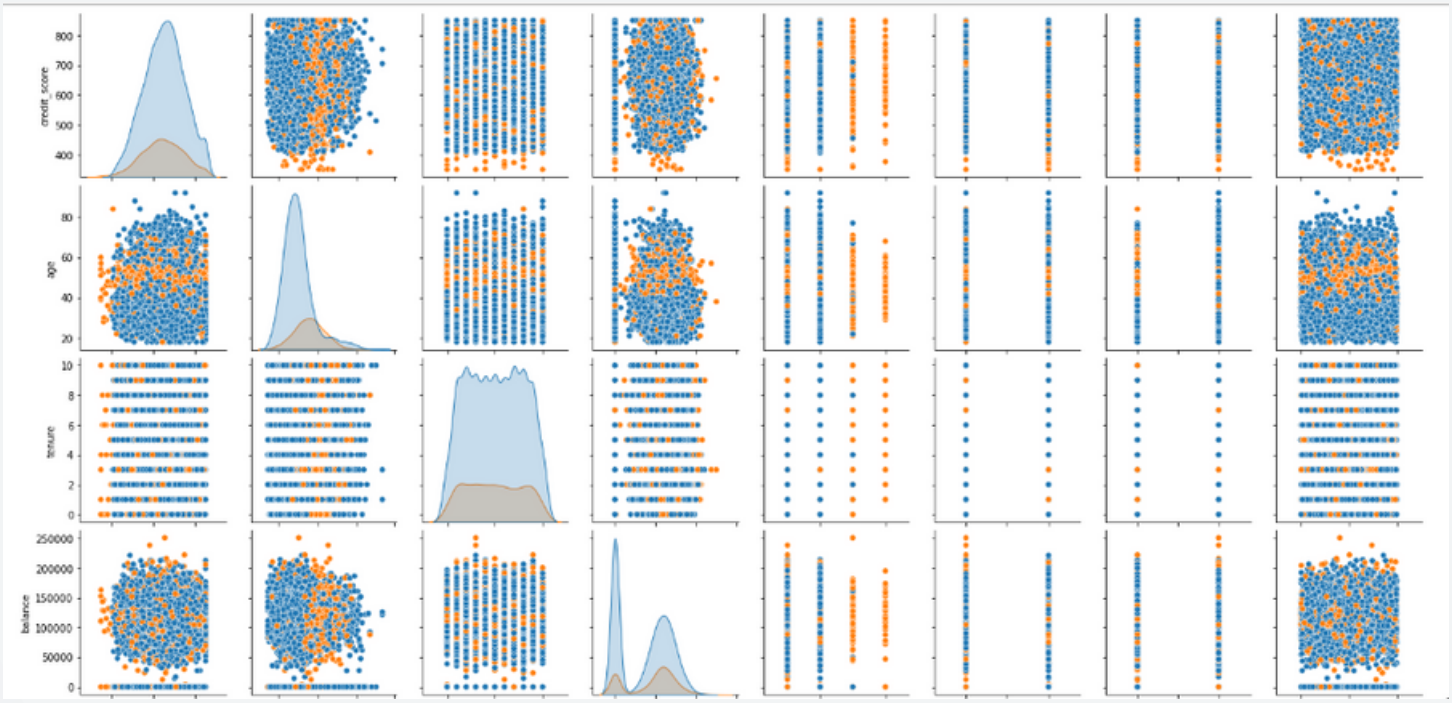

Del analisis exploratorio anterior podemos observar que la mayoria de los registros pertenecen a Francia, teniendo una cantidad similar entre España y Alemania, hay mas registros de hombres, una distribucion bastante pareja de la tenencia, la mayoria tiene entre 1 y 2 productos y si tienen tarjetas de credito. Tambien se observa una distribucion similar entre los clientes activos y los que no, la mayoria de los usuarios tienen entre 20 y 40 años y por otro lado vemos que hay mas clientes que no han salido del banco con respecto a los que si se fueron.

Vamos a ver como se correlacionan nuestras variables




Podemos observar que con relacion a nuestra variable target existe una mayor correlacion con la edad

Realizamos un analisis de la correlacion entre las distintas dimensiones para observar si rapidamente podemos detectar algun insight




Rapidamente no encontramos una combinacion de dimensiones que sea explicativa

# 3 MODELOS NO SUPERVISADOS

En esta parte del analisis lo que realizamos es todo el proceso para obtener un dataset prolijo a nivel datos y escala, ya que por un lado
tenemos variables categoricas que debemos transformarlas a numericas para que podamos utilizarlas en nuestro modelo y por otro lado
cada una de nuestras variables representa una dimensión, un eje, dentro del espacio multidimensional donde se encuentran las observaciones.
No todas se encuentran en la misma escala, por eso es sumamente importante estandarizarlas para eliminar sus distintas unidades de medida y evitar distorsiones debidas a diversas escalas. Tambien pudimos observar a partir del primer analisis exploratorio de los datos, que los mismos no se encuentran balanceados, lo cual nos va a generar distorsiones a la hora de entrenar el modelo, por lo cual en este momento tambien realizamos un balanceo de los mismos a partir de distintas maneras para arribar a la optima.

**Modelos de balanceo de datos (Undersampling - Submuestreo):**

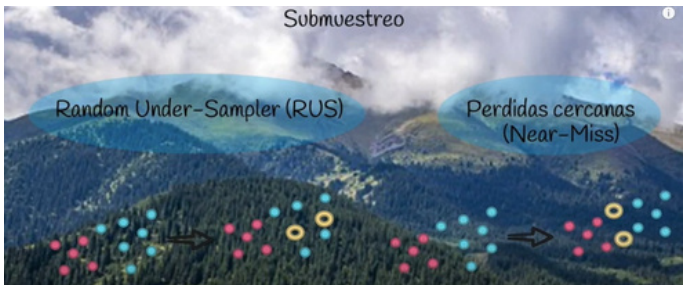
Estos modelos trabajan eliminando observaciones de la variable mayoritaria para equilibrar las muestras.
Antes del balanceo, nuestros datos de la variable target eran los siguientes:

| | |
|---|---|
| **0.** 0.796 | **1.** 0.204 |

Utilizamos: **Random Under Sampler,** Este método de balanceo saca puntos de la clase mayoritaria de manera aleatoria y **Near Miss,** que elimina los registros de la clase mayoritaria más cercanos a la clase minoritaria.
Estos modelos trabajan eliminando observaciones de la clase mayoritaria para equilibrar las muestras
Utilizando una u otra opcion obtenemos los siguientes valores:

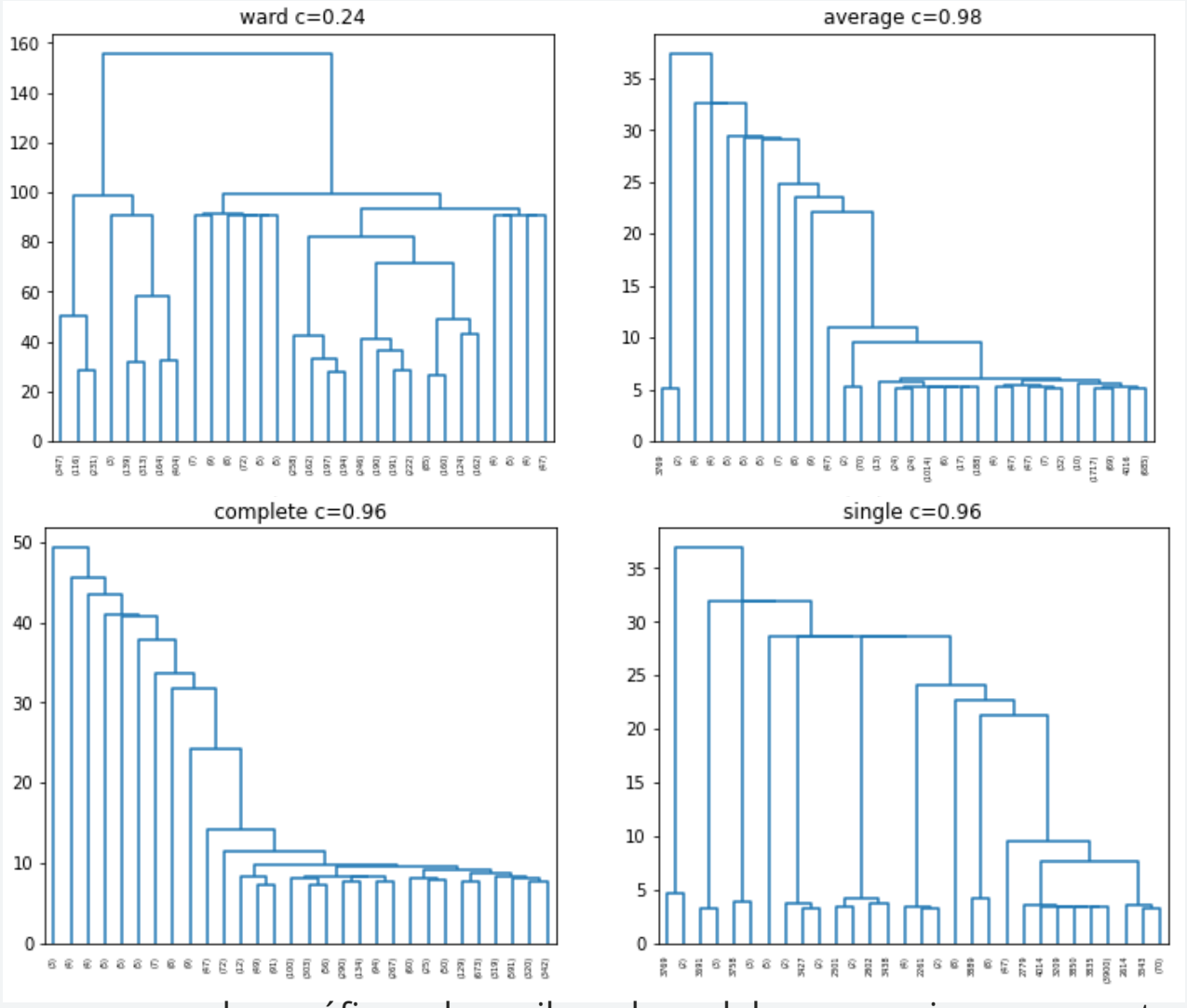
| | |
|---|---|
| **0.** 0.5 | **1.** 0.5 |

A partir de los datos balanceados aplicamos distintos modelos no supervisados para detectar algun cluster significativo para utilizar en nuestro modelo

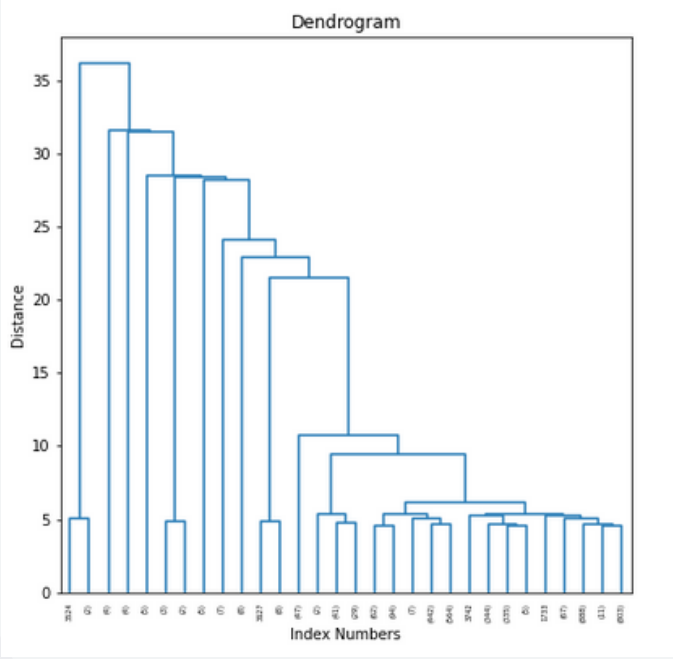


# 3.1) Clustering Jerarquico

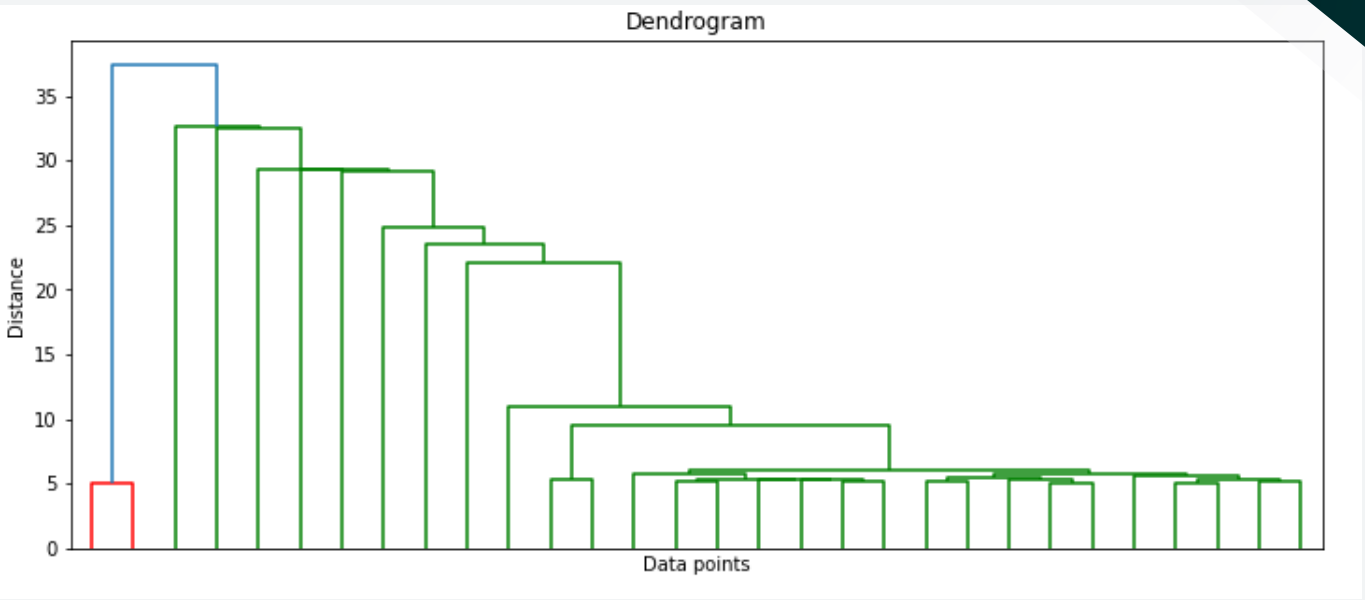
Con este modelo de algortimo de clúster jerárquico lo que hacemos es agrupar los datos basándose en la distancia entre cada uno y buscando que los datos que están dentro de un clúster sean los más similares entre sí




Como vemos en los gráficos de arriba, el modelo que mejor representa los datos en el dendrograma es "AVERAGE", por lo que ploteamos los datos y trabajamos con ese modelo:

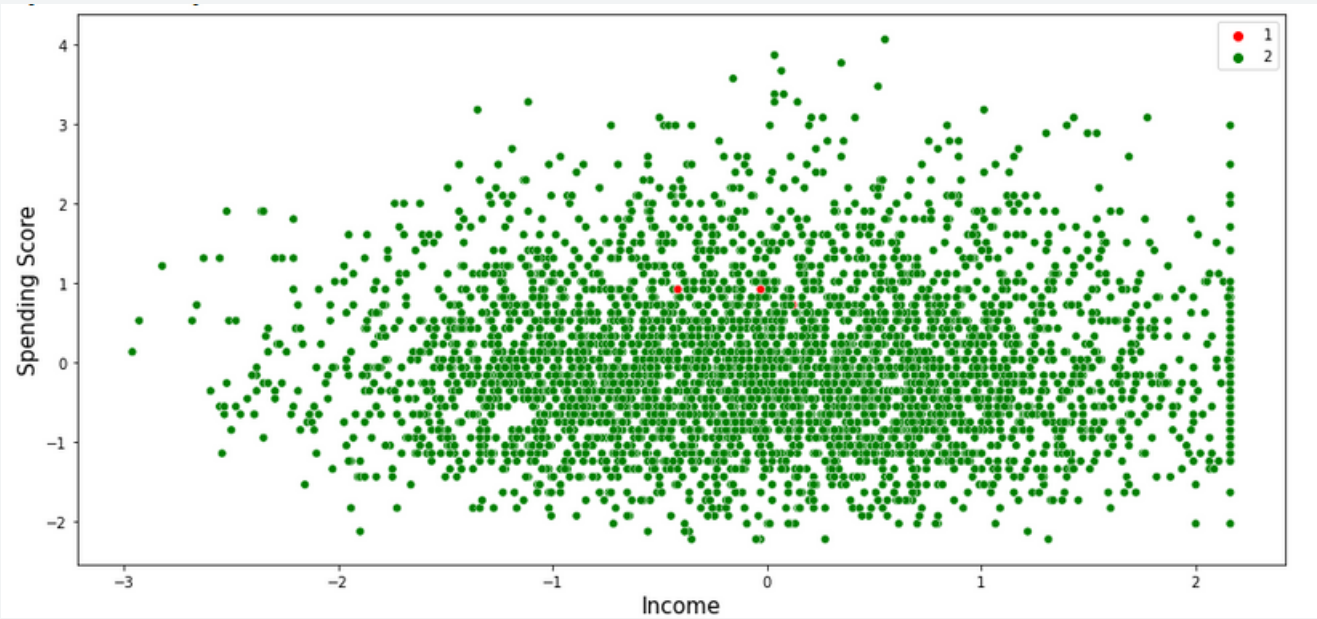



El mejor **coef cofenético** es el que resulta de trabajar con la medida de distancias **Average**, adecuado para trabajar con clusters con distintas cantidades de datos y el valor obtenido es de **0.98**



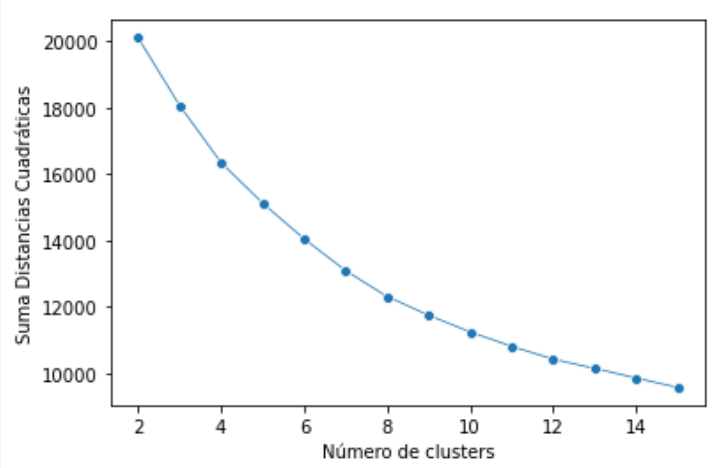

De esta manera quedan definidos 5 clusters, cada uno de los cuales tiene un "diámetro" característico cercano a 4, y que la distancia mínima entre ellos es cercana a 10. Por este motivo esta elección de número de clusters parece adecuada.

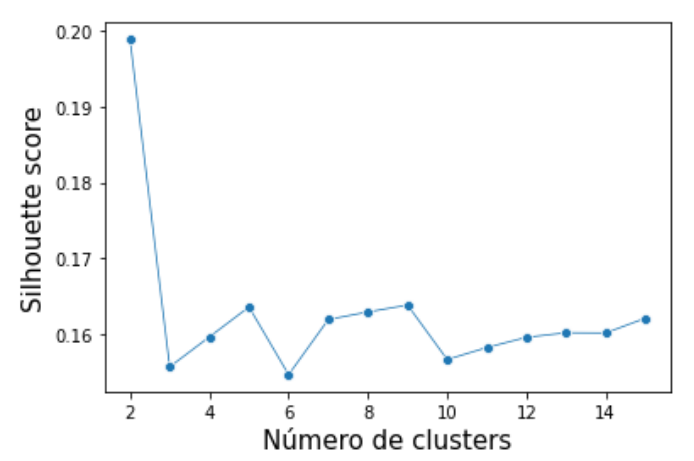
Para ver la composición exacta de cada cluster usamos la función fcluster, pasando como argumento la distancia de corte o directamente el número de clusters

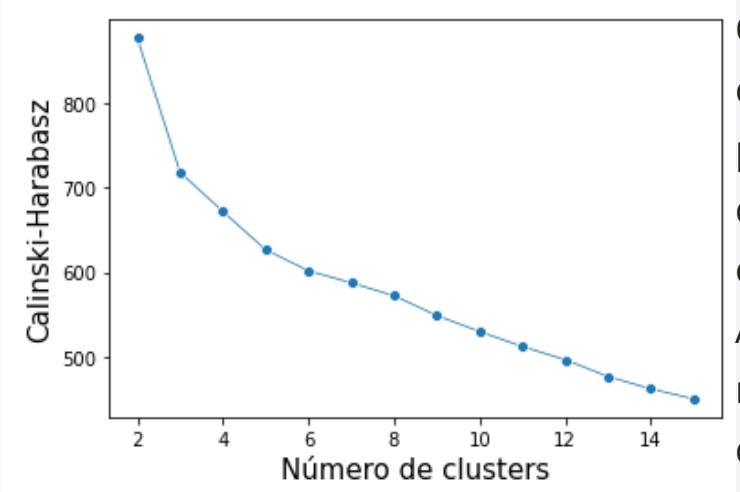


## 3.2) K-Means

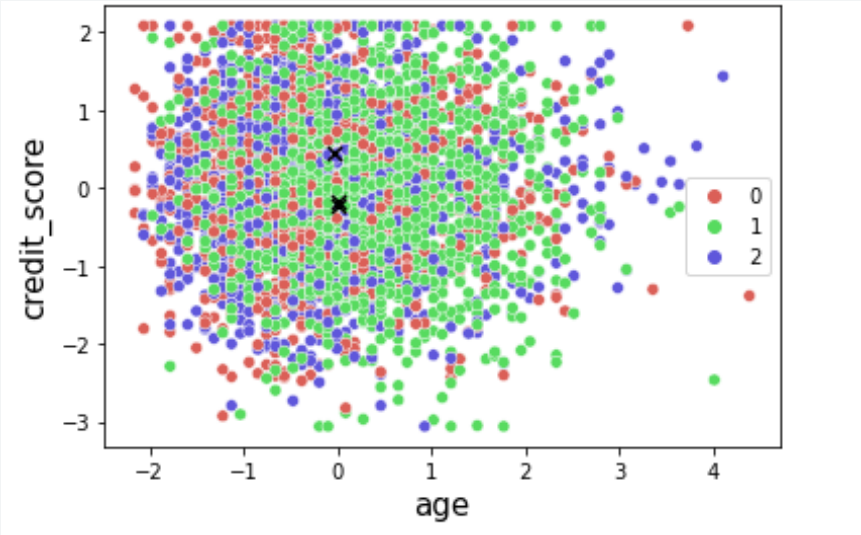
Usamos K-means para ver a priori si existe alguna distribución de datos que representen los clusters que estamos buscando.
Para ello calculamos el número óptimo de K clusters a travez de diferentes coeficientes como ser la suma de distancias al cuadrado, el coeficiente de Shilouette y el coeficiente de Calinski Harabasz

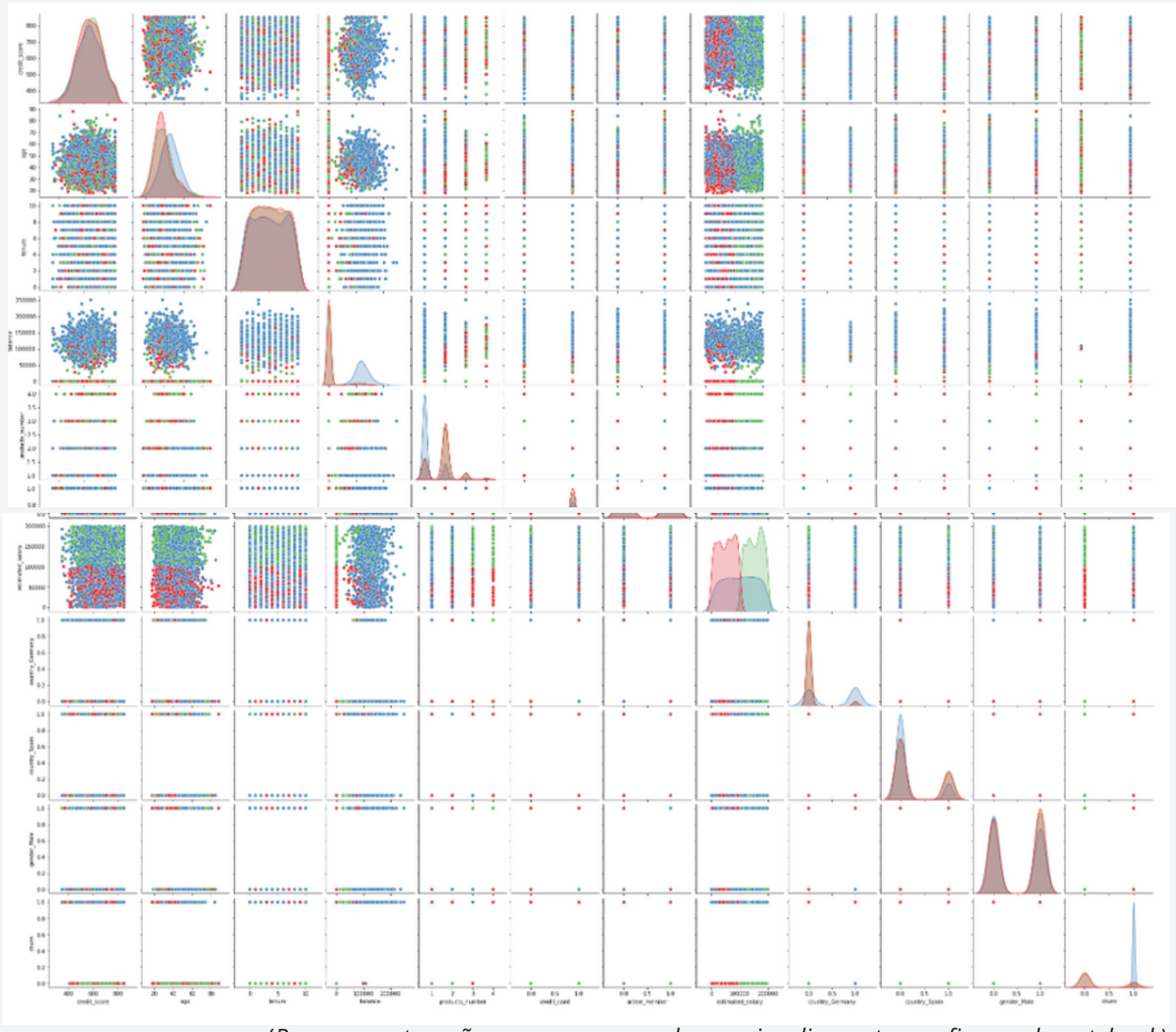



Observamos que 3 es un número de clusters recomendado, lo cual podemos interpretar como 0: se van del banco, 1: están indecisos si se van o no, 2: no se van del banco.
Ahora procedemos a representar el resultado en un gráfico para ver los distintos clusters





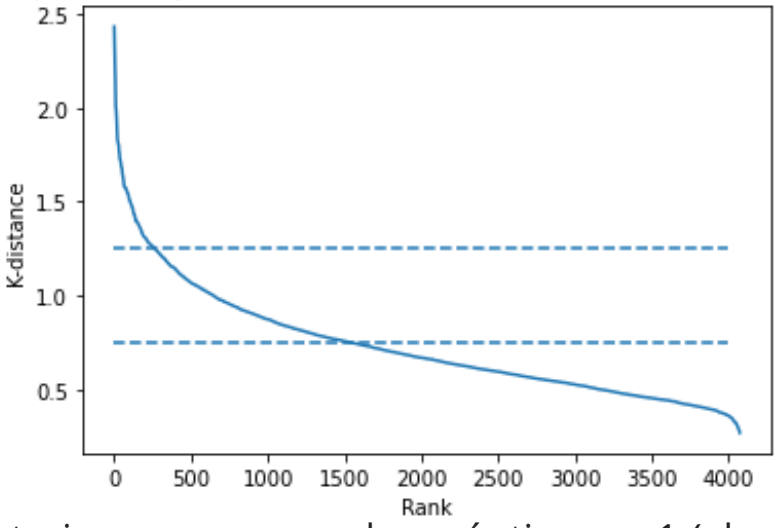
(*Para ver en tamaño mayor, recomendamos visualizar estos graficos en la notebook)*

En este caso, la representación gráfica a traves del método no supervisado K-means, no nos ayuda a comprender mejor nuestra predicción

# 3.3) DBSCAN

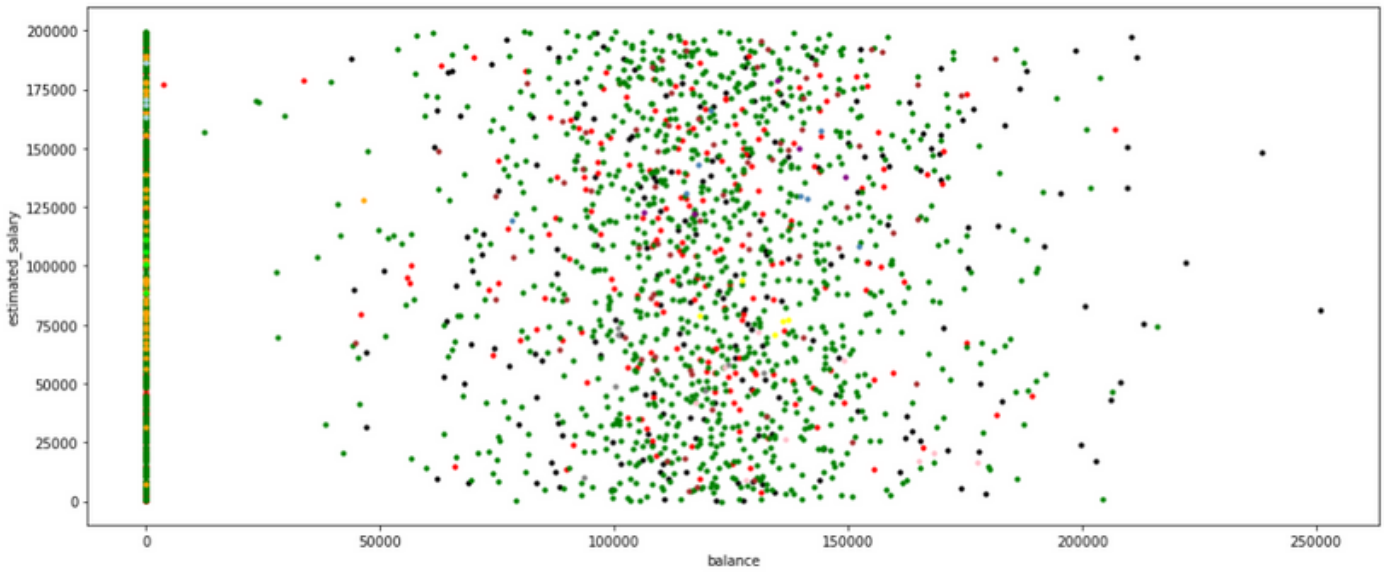
Utilizamos DBSCAN para detectar en forma automatica cluster, patrones de agrupacion en el espacio fisico y detectar Outliers.

Analizamos el epsilon óptimo

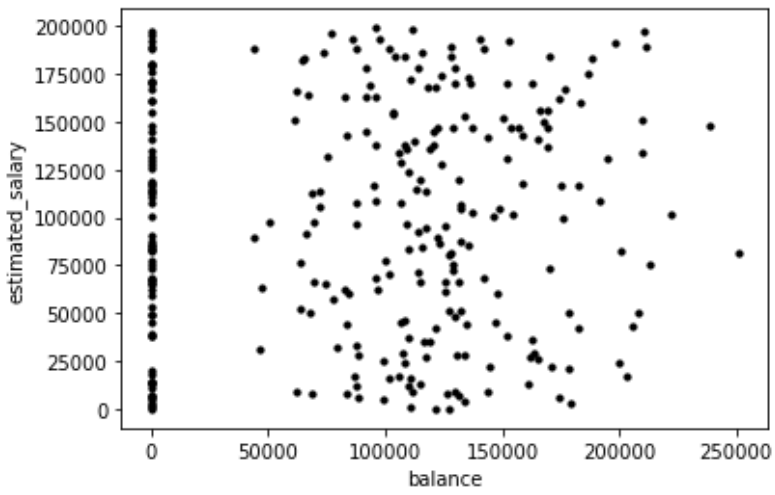


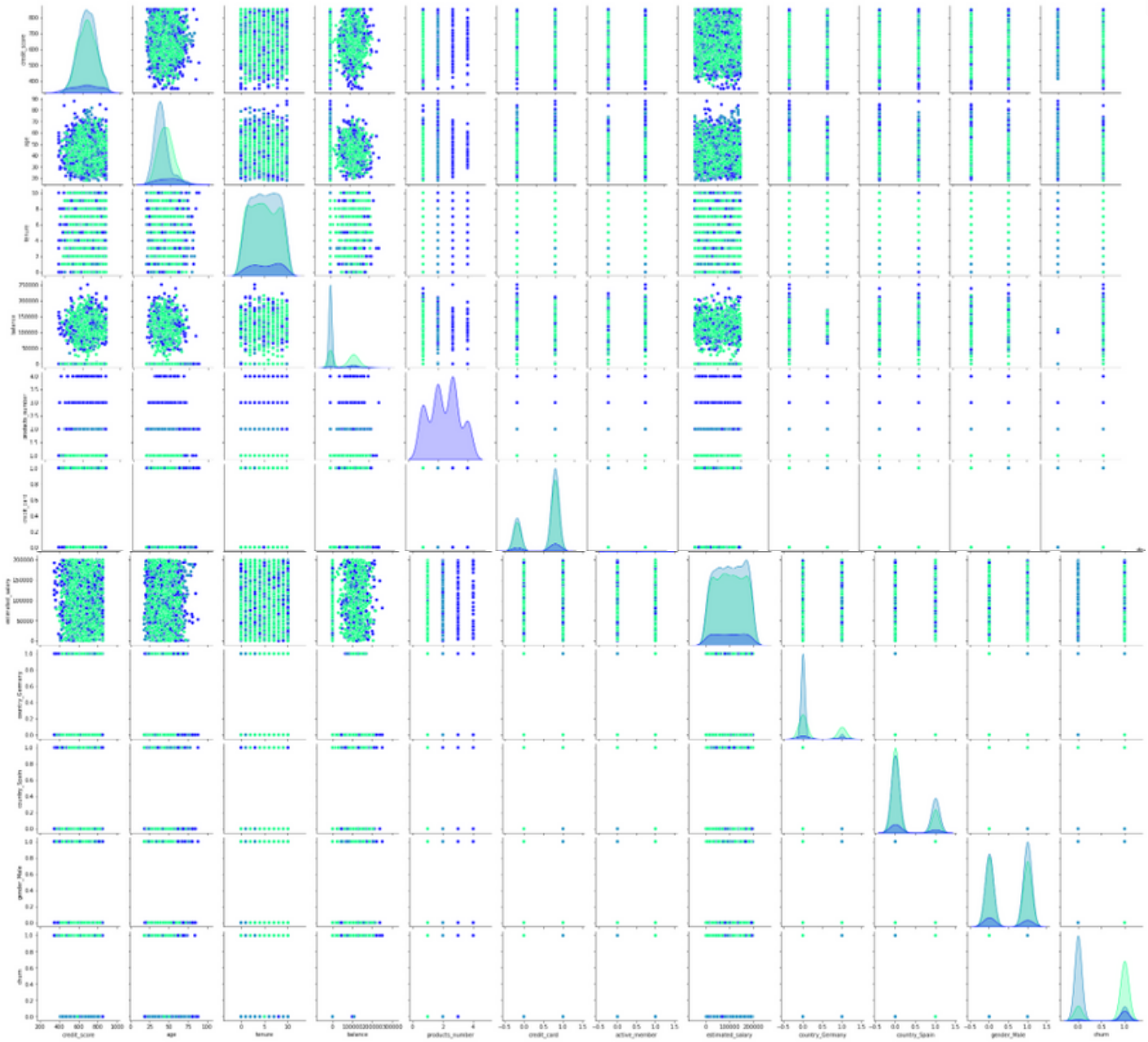

Del gráfico anterior, vemos que el eps óptimo es 1 (el comprendido entre las dos líneas punteadas).

Observamos los primeros 10 clusters:




Ploteamos los outliers para tener una clara idea de su ubicación





(*Para ver en tamaño mayor, recomendamos visualizar estos graficos en la notebook)*

A partir de este analisis vamos a entrenar distintos modelos de clasificacion, agregando al mismo las variables dummies obtenidas en los modelos de K-Means/DBSCAN y ambas, sacando o dejando los outliers y analizando los valores obtenidos; en la notebook se puede observar los valores de dischos modelos utilizando las distintas opciones, en esta presentacion mostramos los resultados optimos obtenidos a partir de utilizar la combinacion de las variables dummies de ambos modelos y sin eliminar los outliers.

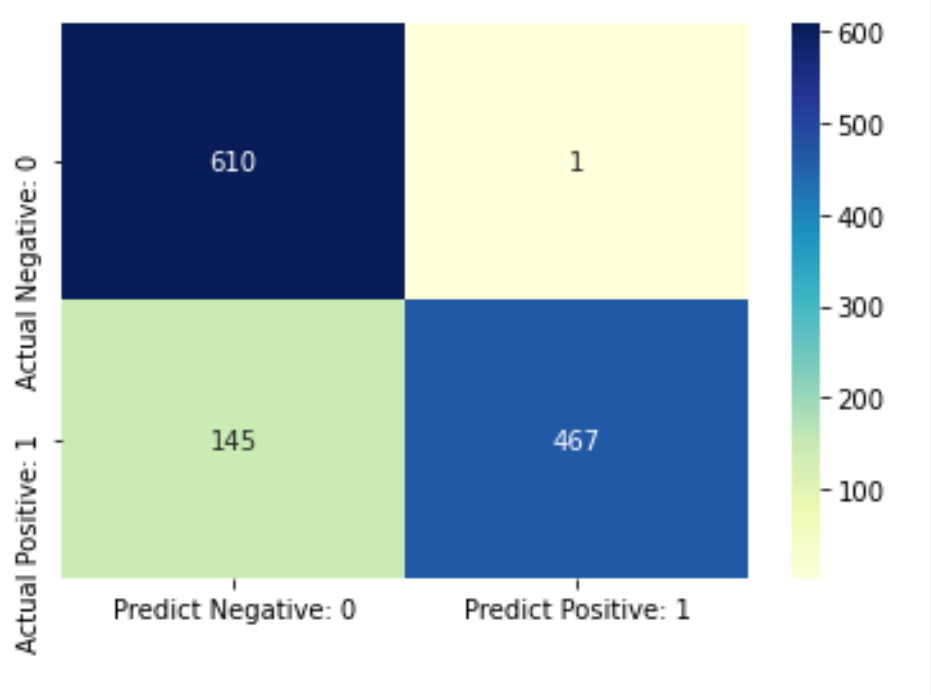
# 4 MODELOS DE CLASIFICACION

## 4.1) Naive Bayes

Recordemos que el Modelo de Naive Bayes no tiene Hiperparámetros, por lo que no utilizaremos Grid Search para este Modelo.
Los valores de nuestro y train o variable target al inciar el modelo es de :

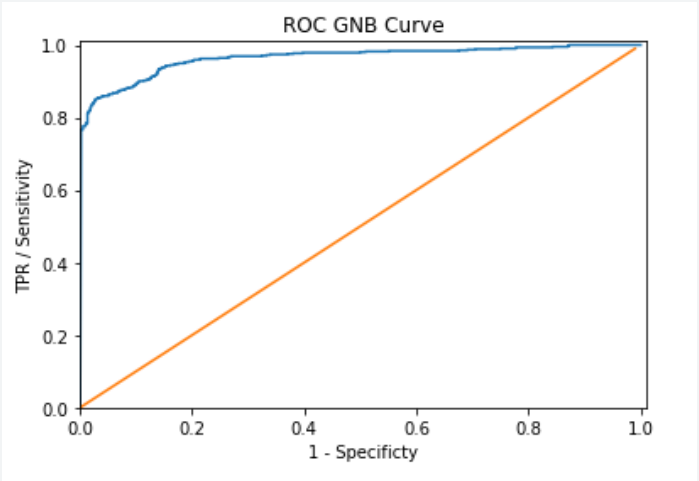
**0.** 0.50 **1.** 0.50

Evaluamos el modelo y obtenemos un **Recall** de **0.763** y la sig. matriz de confusion:




Como podemos ver, al usar un df desbalanceado, predice muy bien los casos negativos, es decir que abandonan el banco ya que es la predicción para la cual tiene mayor cantidad de datos.
Luego observamos las principales metricas para evaluar nuestro modelo y la curva ROC que nos arroja el mismo:

```
Accuracy bayes = 0.881

Recall bayes = 0.763

Precision bayes = 0.998

Specificity bayes = 0.998

F1 Score bayes = 0.865
```



# 4.2) KNN

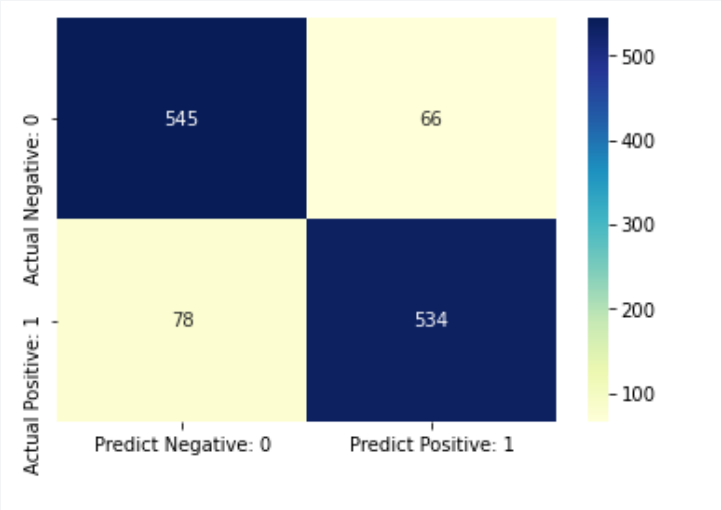
Esta es una técnica simple de clasificación conocida como k_Nearest Neighbors (KNN), (vecinos más cercanos).

Básicamente, los pasos de KNN son:

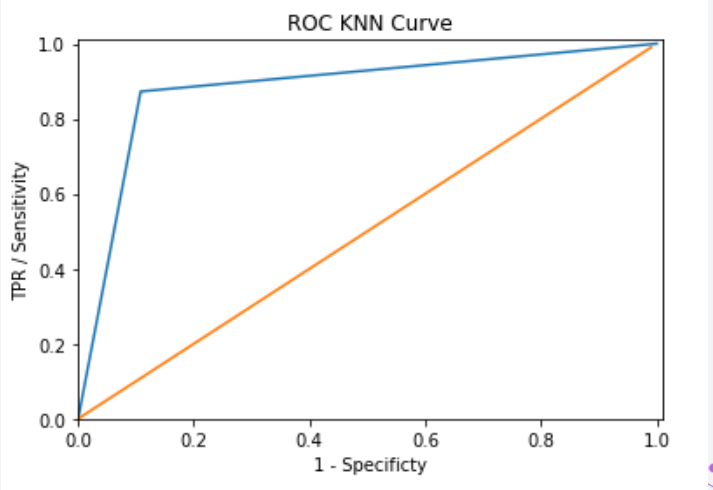
1. Memoriza la ubicación de cada observación del train dataset, según los valores de sus _features
2. Cuando recibe un dato nuevo, lo ubica también en el espacio.
3. Encuentra los _k_ vecinos más cercanos (k es un hiperparámetro del modelo).
4. Cada uno de ellos aporta un "voto" a la clase a la que pertenece.
5. La predicción para el dato nuevo queda determinada por la clase mayoritaria entre los "votos" del paso anterior.

En una primer prueba, trabajaremos con la configuración de los hiperparámetros del modelo que viene por defecto y luego para obtener la combinación de hiperparámetros óptima, trabajamos con grid search.

Obtuvimos un **Recall de 0.873** con la siguiente matriz de confusion, los valores de las principales metricas y la curva roc:

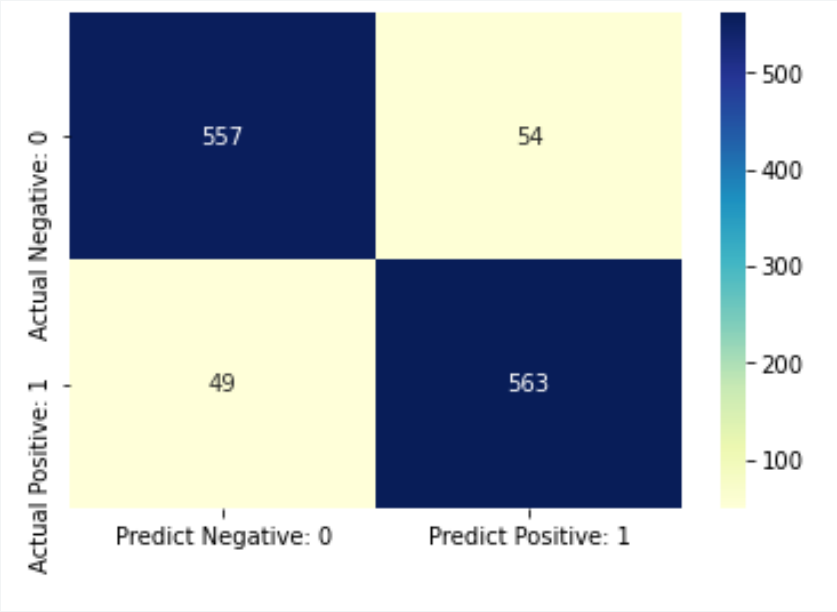



```
Accuracy knn = 0.882

Recall knn = 0.873

Precision knn = 0.89

Specificity knn = 0.892

F1 Score knn = 0.881
```

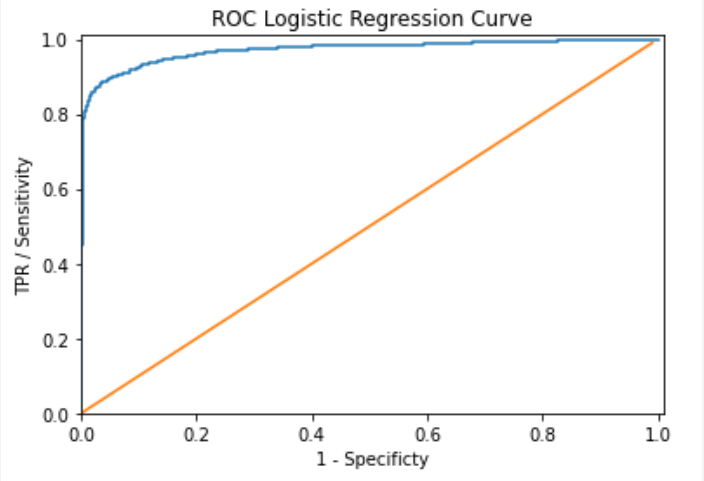


# 4.3) Regresion Logistica

Utilizamos el modelo de Regresion Logistica para determinar la probabilidad de que ocurra un evento. Este modelo nos muestra la relación entre las características y luego calcula la probabilidad de un resultado determinado.
Buscamos la mejor combinacion de hiperparametros con GridSearchCV y obtuvimos un **Recall de 0.92**, con la siguiente matriz de correlacion, valores principales y curva ROC:



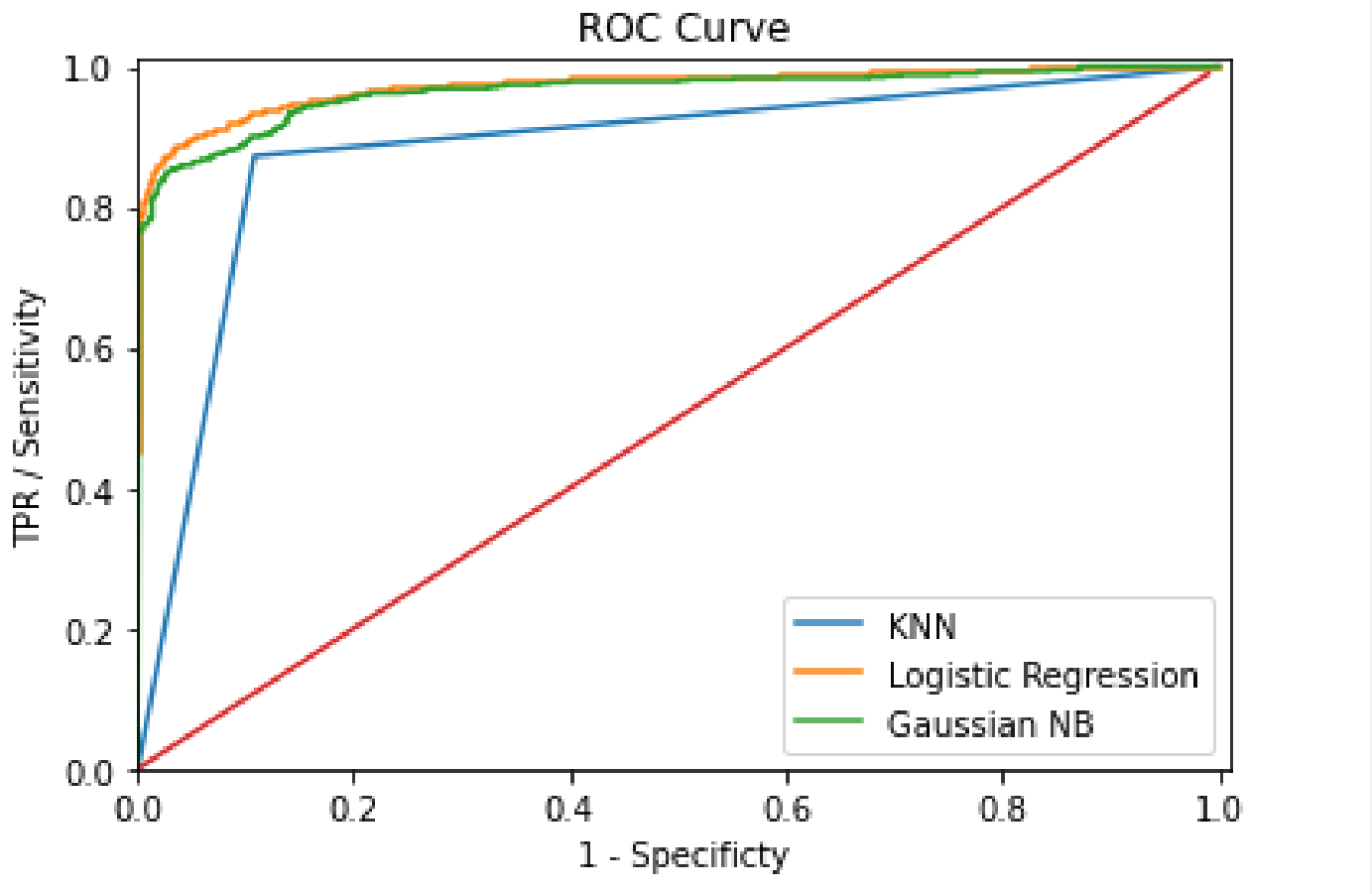

```
Accuracy Lg Rg = 0.916

Recall Lg Rg = 0.92

Precision Lg Rg = 0.912

Specificity Lg Rg = 0.912

F1 Score Lg Rg = 0.916
```



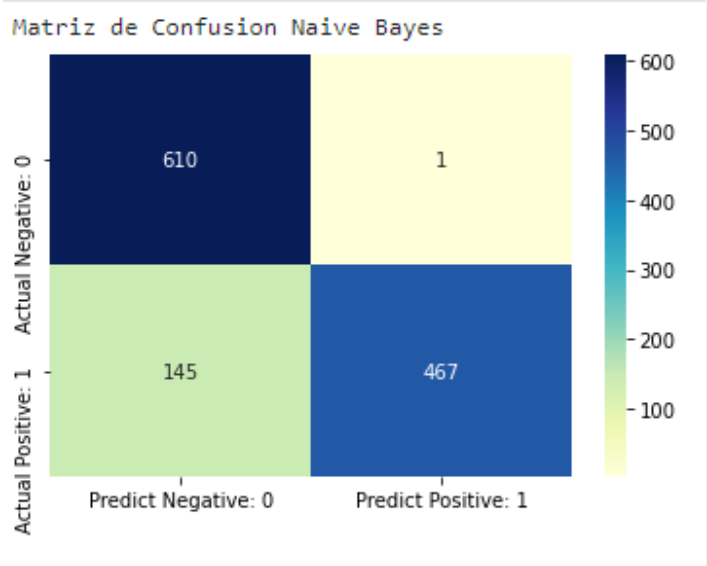
# 5 CONCLUSIONES

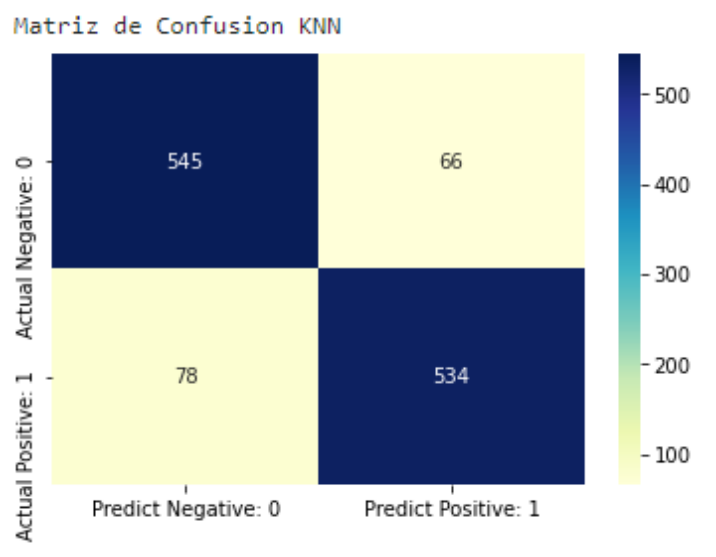
En una primera instancia comparamos las principales metricas y las curvas ROC obtenidas a partir de cada uno de los distintos modelos aplicados:

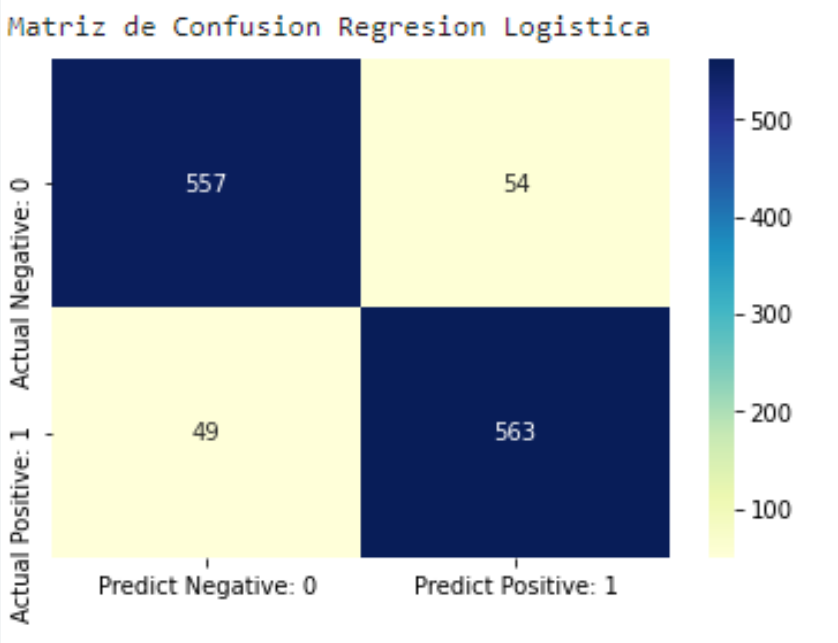
| | Model | Accuracy | Recall | Precision | Specificity | F1 Score | AUC |
|---|---|---|---|---|---|---|---|
| **1** | Regresión Logistica | 0.916 | 0.920 | 0.912 | 0.912 | 0.916 | 0.971476 |
| **0** | KNN | 0.882 | 0.873 | 0.890 | 0.892 | 0.881 | 0.882265 |
| **2** | Gaussian Naive Bayes | 0.881 | 0.763 | 0.998 | 0.998 | 0.865 | 0.964405 |



Luego observamos tambien las matrices de confusion obtenidas de cada modelo:




**A partir de todo el analisis realizado podemos concluir que el Modelo de Regresion Logistica es el que mejor predice los resultados True Positive, es decir, en este caso, predice mas eficientemente los clientes que se van a ir del banco, lo cual tambien lo podemos observar en los valores obtenidos en las distintas metricas para evaluar los modelos, el que mejores valores nos arroja es el de Regresion Logistica, a su vez tambien podemos observar que minimiza el error de predecir mal cuando un cliente en realidad se va pero lo clasificamos como negativo.**

**En Resumen, concluimos que el modelo de Regresión Logística es el que mejor nos sirve para nuestro objetivo de predecir el Churn, ya que es el modelo que maximiza el Recall (Sensitivity), es decir maximiza los TP (predice más eficientemente a los clientes que se van a ir del banco) y minimiza a los FN (minimiza el error de predecir mal cuando un cliente en realidad se va del banco, pero nosotros predecimos que se quedaba). Además, vemos que el Modelo de Regresión Logística tiene la mayor AUC, lo cual nos da una idea general de cómo performa el modelo, y nos indica que el modelo va a ser mas robusto a cambios de thresholds**