**Description**

Your company wishes to better understand the evolution of our sales globally.

More specifically, it is unclear whether we should create a VIP program for our clients, focus on particular customer segments or which markets, if any, we should pay more attention to. We also do not know if higher category shipments are being profitable or we are losing money instead and should drop them.

Try to help your company improve its situation.

**Submit a pdf file** applying machine learning techniques to try and extract insights from the sales data. For each technique, you should include

1. a screenshot of the implementation in jupyter
2. a visualization of the model or its results (if it is possible)
3. an interpretation of the results obtained by the model (accuracy, coefficients)

Remember that the file submitted must include the full name(s) of the student(s) involved.

**To do**:

Using the data from the previous session OR from the ORACLE database, add a new column ‘VIP’ with two values: 0 and 1. The value for VIP will be 1 for those in the list of top 50 customers according to the profit made thanks to them, 0 for the rest.

Train and interpret the output of the following models:

* Decision Tree with target to predict: VIP
* Decision Tree Regressor with target to predict: Profit
* Linear Regression with target to predict: Profit
* Kmeans with scaler

# Preparación de datos

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Ilustración 1

1. Hacemos la conexión en la base de datos
2. Obtenemos los 50 mejores clientes con la consulta de SQL.

Texto, Carta

Descripción generada automáticamente

Ilustración 2

Obtenemos los registros de los pedidos. Obtenemos el mercado, la categoría del producto, el mes y año de pedido y de envío y las medidas del fact (*profit, shipment cost, sales, quantity*). Además, tendremos los *ids* de prioridad, *shipment y returned* que no los obtenemos el *SQL* porque nos interesa tener valores no numéricos para los entrenamientos.

Además, insertamos el atributo VIP.

Texto

Descripción generada automáticamente

Ilustración 3

Le damos un valor numérico cada mercado y un valor numérico a cada producto de categoría. Luego eliminamos los ids que no nos sirven.

Pantalla de computadora

Descripción generada automáticamente con confianza media

Ilustración 4 Dataframe con los resultados

# Decision Tree with target to predict: VIP

Texto

Descripción generada automáticamente con confianza media

Ilustración 5 Report con ydata\_profilng

Imagen que contiene Interfaz de usuario gráfica

Descripción generada automáticamente

Ilustración 6

Como podemos ver solo el 0.7% de los pedidos provienen de clientes vips lo que va a causar muchas dificultades para entrenar un modelo que nos sea útil.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Ilustración 7

De hecho, si entrenamos un clasificador veremos que la precisión es casi 1 por lo que puede ser que no nos sea de mucha utilidad. El clasificador podría tener esa gran precisión ya que como hemos visto la mayoría de los pedidos no provienen de clientes VIP. Si nos pusiéramos en la situación de que el clasificador responde siempre a que el pedido no es VIP, teniendo en cuenta que solo un 0.7% de predidos son VIP aun así obtendría una precisión de 0.993 (como vemos coincide con la precisión del clasificador).

Lo que podríamos hacer es obtener todos los pedidos VIPS y obtener al azar el mismo número de pedidos no VIPS y probar a entrenar el clasificador.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Ilustración 8

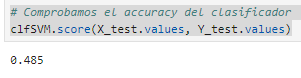


Ilustración 9

Volvemos a entrenar el clasificador y obtenemos 0.485. Es un valor de precisión muy bajo para sacar alguna conclusión interesante mediante un árbol de decisión. Es posible que el valor sea bajo la precisión porque el total de pedidos en el *dataset* es tan solo de 604 pedidos, es demasiado pequeño.

Si usáramos el clasificado entrenado con todos los datos, aunque probablemente esté muy lejos de ser ideal. Podríamos crear el árbol de decisión de la siguiente manera:

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Ilustración 10

De esta forma podríamos obtener el árbol de decisión. En mi caso he puesto 3 niveles de profundidad.

Diagrama

Descripción generada automáticamente

Ilustración 11

Según este calificador lo más probable es que un pedido sea de un cliente vip si tiene un profit entre **1138.02 y 1814.482**.

Si obtuviéramos el árbol de decisión para el otro calificador con los datos balanceados de los pedidos con un 50 % vip y 50% no vip(aunque es poco preciso) obtendríamos esta bondad del clasificador y este diagrama:

Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

Descripción generada automáticamente

Ilustración 12

Diagrama

Descripción generada automáticamente

Ilustración 13

Según este árbol de decisión

* Clientes con MONTH\_SHIP > 2.5 (es decir de Marzo en adelante) y SHIPPING\_COST <= 11.523 tienen alta probabilidad de ser VIP.
* Clientes con QUANTITY <= 1.5 también tienden a ser VIP.
* Clientes con PROFIT > 1042.677 pueden ser VIP, pero son casos raros.
* Los clientes con DISCOUNT <= 0.085 y PROFIT bajo tienen menos probabilidades de ser VIP.

# Decision Tree Regressor with target to predict: Profit

Esta vez vamos a hacer un árbol de regresión para intentar predecir el *profit*. Se entrena el modelo de regresión y especificamos una profundidad de 2 en este caso.

Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

Descripción generada automáticamente

Ilustración 14

Podemos gráfica el resultado de la siguiente manera:

Diagrama

Descripción generada automáticamente

Ilustración 15

El modelo encuentra que los beneficios son generalmente más altos cuando *sales* es bajo y *discount* es bajo. Sin embargo, en algunos casos con altos descuentos (DISCOUNT > 0.485), los beneficios pueden volverse negativos. Las muestras con SALES > 1812.03 son pocas (851 de 46000), pero presentan una gran variabilidad en *profit*, con algunas pérdidas significativas. Un descuento mayor a 0.285 parece estar asociado a pérdidas cuando *sales* es bajo.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Ilustración 16

Como vemos el coeficiente muy pequeño, Un valor de 0.317 indica que el modelo captura cierta relación entre las variables, pero queda un 68.27% de la variabilidad de los datos sin explicar.

El modelo no es muy preciso, aunque tampoco inútil, probablemente por la profundidad del árbol. Si probamos con un árbol de profundidad 3 obtendríamos lo siguiente:

Diagrama, Esquemático

Descripción generada automáticamente

Ilustración 17 Árbol de regresión de 3 profundidades

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

Ilustración 18 Nuevo coeficiente calculado

También podríamos ver los pesos de la importancia para cada variable de la siguiente forma:

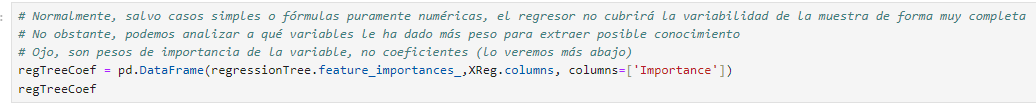


Ilustración 19 Importancia para árbol de regresión de profundidad 2

Tabla

Descripción generada automáticamente

Ilustración 20 Importancia de cada variable en los árboles de regresión profundidad 2 y 3

Las únicas variables importantes que son importantes parecen ser *sales* y *discount.*

# Linear Regression with target to predict: Profit

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Ilustración 21

El modelo de regresión lineal entrenado muestra un coeficiente de determinación 0.0469, lo que indica que solo explica el 4.69% de la variabilidad en los datos, sugiriendo que nuevo es un ajuste pobre.

Interfaz de usuario gráfica, Tabla

Descripción generada automáticamente con confianza media

Ilustración 22

Los coeficientes muestran el impacto de cada variable en la predicción, donde DISCOUNT (0.009443) tiene un efecto positivo y YEAR\_ORDER (-0.025846) uno negativo, entre otros. Lo más probable que debido al bajo desempeño del modelo la relación entre variables no sea lineal.

# Kmeans with scaler

Antes de empezar a agrupar por clusters, tenemos que elegir el nº de clusters. Voy a basarme en dos conceptos:

1. El coeficiente de silhouette que mide la separación entre clusters (valores más altos indican mejor agrupación).
2. El método del codo analiza la “inercia” (suma de distancias cuadradas de los puntos al centroide de su cluster). La idea es elegir un número de clusters donde la reducción de la inercia deje de ser significativa (el "codo" de la curva).

Lo hacemos de la siguiente manera:

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Ilustración 23

Vamos calculando la inercia y coeficiente de silhouette para cada nº de clusters.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Ilustración 24 Visualización utilizando seaborn y matplotlib

Gráfico, Gráfico de líneas

Descripción generada automáticamente

Ilustración 25 Resultado

Basándonos en ambos métodos parece que 4 o 5 son las mejores opciones para separar los clusters, en mi caso voy a usar 4.

Imagen que contiene Tabla

Descripción generada automáticamente

Ilustración 26

Otengo los clusters y se insertan en un *dataframe.*

Gráfico, Gráfico de dispersión

Descripción generada automáticamente

Ilustración 27

Lo visualizamos con un *bubble plot.* A continuación, lo comparamos con los clusters obtenidos tras realizar un **escalado**.

Tabla

Descripción generada automáticamente con confianza media

Ilustración 28

Kmeans es sensible a la magnitud de los datos, por lo que se normalizan todas las variables para que tengan media 0 y desviación estándar 1. Seguidamente volvemos a calcular los clusters y los volvemos visualizar con un *bubble plot*.

Gráfico, Gráfico de dispersión

Descripción generada automáticamente

Ilustración 29 Visualización final del Kmeans escalado