# **Índice**

[**Índice**](#_q6r20ijyl0zo) **1**

[**Integrantes del grupo**](#_dzxnbl9emgve) **2**

[**Descripción del problema**](#_qzvvtms2x6xa) **2**

[Problema Real](#_jmkcs8ws2fan) 2

[Resolución del Problema](#_jx01a7jo7oir) 2

[**Librerías empleadas**](#_g7o9uw6xcscc) **2**

[**Proceso**](#_2swu8dwb0mas) **3**

[Análisis Exploratorio de Datos](#_ixl3b1j95m3j) 3

[DecisionTree](#_la5yyqjoktmt) 7

[RandomForest](#_ws529mrb78vb) 9

[SVM](#_dgh1m5fxyltr) 9

[Naive-Bayes](#_9ur9qf21jv9g) 9

[Naive-Bayes Multinomial](#_5yekofiuz4rk) 10

[**Modelo elegido**](#_bxui6djrojnj) **11**

[**Clustering**](#_n6j9kqgn5b9g) **11**

[Elección del número de clusters](#_imjfp5rhmupv) 11

# 

# **Integrantes del grupo**

José Manuel Vega Gradit, Víctor Montesdeoca Fenoy e Iván Eugenio Tello López.

# 

# **Descripción del problema**

## **Problema Real**

Una fábrica se encuentra con el reto de identificar unos cilindros y clasificarlos en dos grandes grupos: metales y rocas. El primero será utilizado en la construcción de un parque de atracciones, mientras que el segundo se utilizará con el propósito de fabricar elementos decorativos.

Para la empresa es de vital importancia minimizar la aparición de falsos positivos en la detección de metales y que utilizar rocas en lugar de metales para la construcción de las atracciones resultaría catastrófico.

## **Resolución del Problema**

Consideramos como positivo que los cilindros sean metálicos. Debido a que los cilindros metálicos se van a usar para la construcción de una atracción, debemos priorizar que no haya falsos positivos ya que estos pondrían en peligro a la gente que se monte en la atracción.

Por tanto buscamos maximizar la **precisión** en la matriz de confusión.

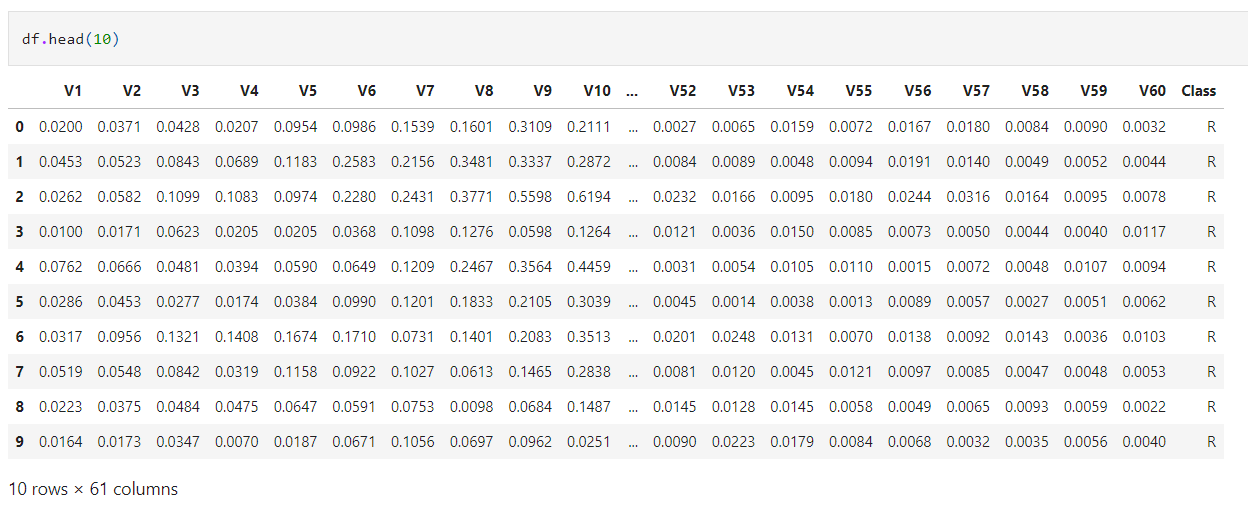
# **Librerías empleadas**

* Pandas: Librería de código abierto de python para la manipulación, análisis y visualización de datos, las características más atractivas que motivan el uso de esta librería son la flexibilidad, la rapidez y lo poderosa que es.
* Plotly: Librería de código abierto de python para realizar gráficas interactivas y de alta calidad.
* Cufflinks: Librería de python que conecta plotly con pandas para crear gráficas directamente sobre data frames.
* Sklearn: Es un conjunto de rutinas escritas en Python para hacer análisis predictivo, que incluyen clasificadores, algoritmos de clusterización, etc. Está basada en NumPy, SciPy y matplotlib.

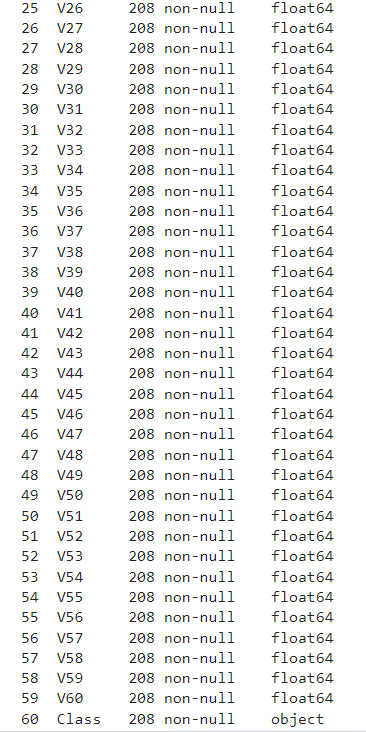
# **Proceso**

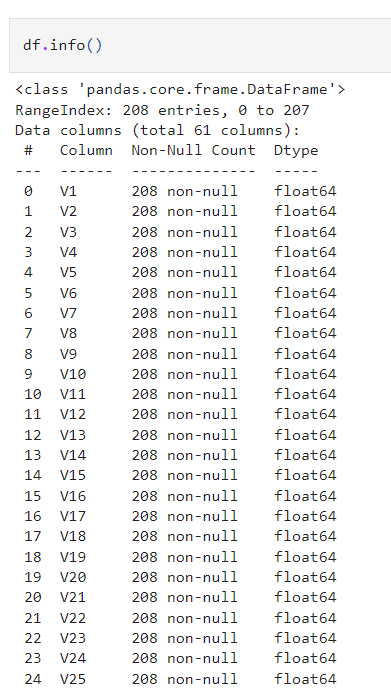
## **Análisis Exploratorio de Datos**

En primer lugar, visualizamos las 10 primeras filas del conjuntos de datos para hacernos a la idea de cómo son los datos.



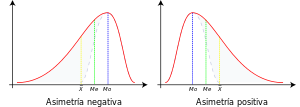
Parece que, salvando la columna de las clases todas las variables son numéricas. Como esto de vital importancia para utilizar los modelos con la librería de Sklearn (no funciona con variables categóricas), nos aseguramos viendo el tipo de datos de cada columna:



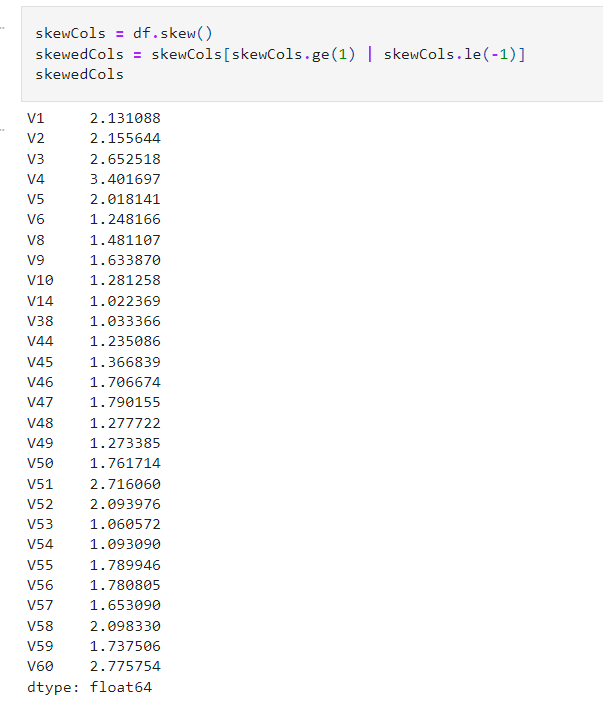


Tras esto, para poder entender mejor los resultados que vamos a obtener y ver si sería necesario normalizar los datos, miramos la distribución de los datos.

Primero, calculamos la métrica de *skewness* o asimetría estadística, que nos dirá la distribución que siguen nuestros datos, para cada columna. Si esta es negativa significará que la distribución es asimétrica positiva o a la derecha. mientras que si es positiva entonces la distribución es asimétrica negativa o a la izquierda. Además, si la asimetría es menor de -1 o mayor de 1, esto significa que los datos siguen una distribución muy asimétrica.



Buscamos las columnas con una asimetría menor que -1 o mayor que 1:



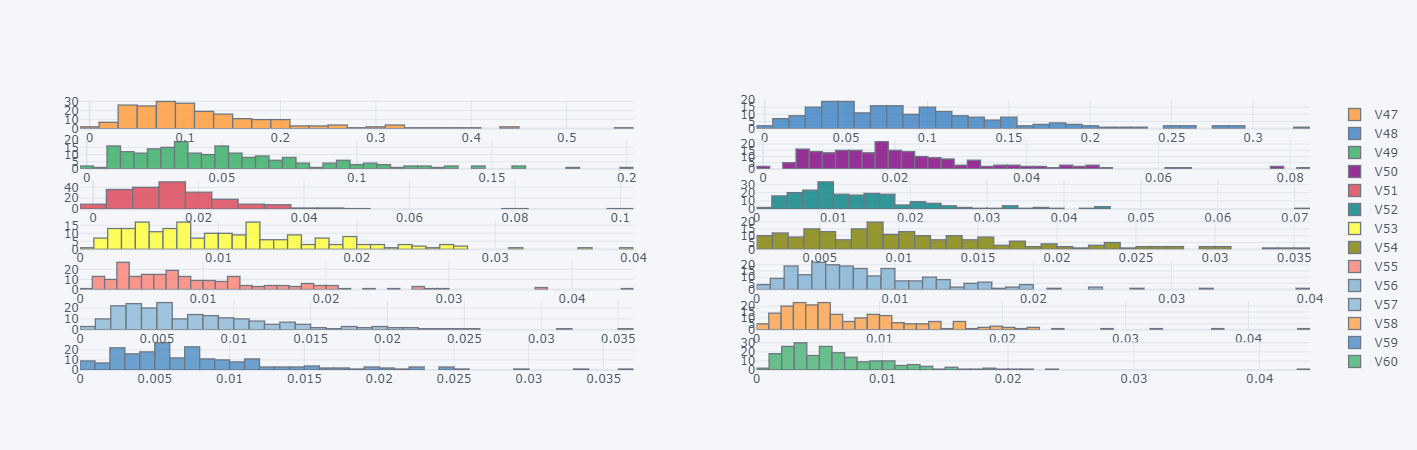
Hay un total de 28 columnas que cumplan esta condición. De manera gráfica, representando mediante histogramas con 50 barras, podmeos observar las distribuciones de cada una de ellas.

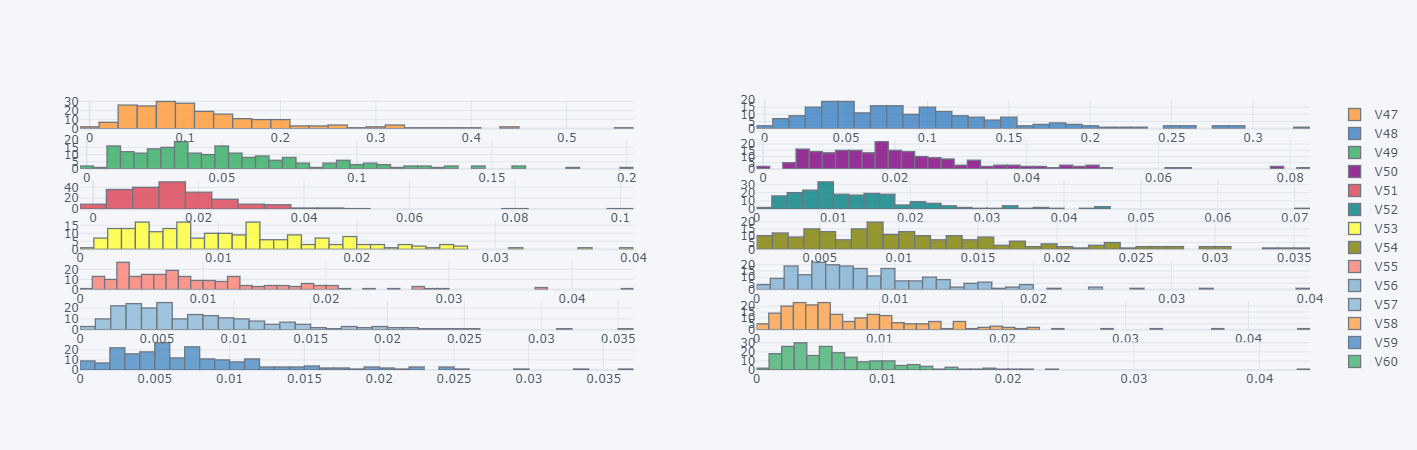
Las primeras 14 columnas:





Las segundas 14:



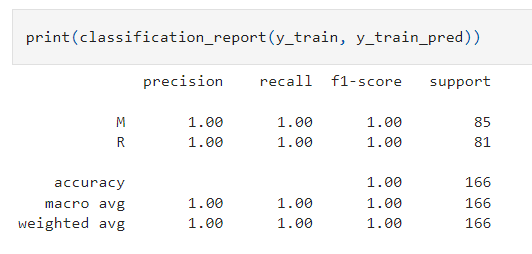


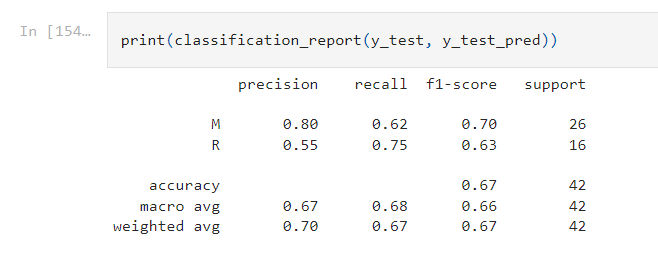
**Estudio de posibles modelos**

Los modelos de aprendizaje automático supervisado que vamos a estudiar son DecisionTree, RandomForest, SVM, Naive-Bayes y Naive-Bayes Multinomial.

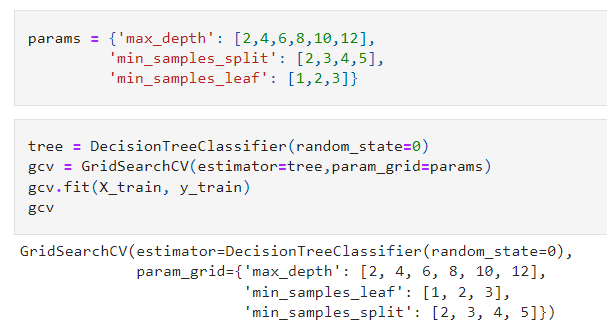
## **DecisionTree**

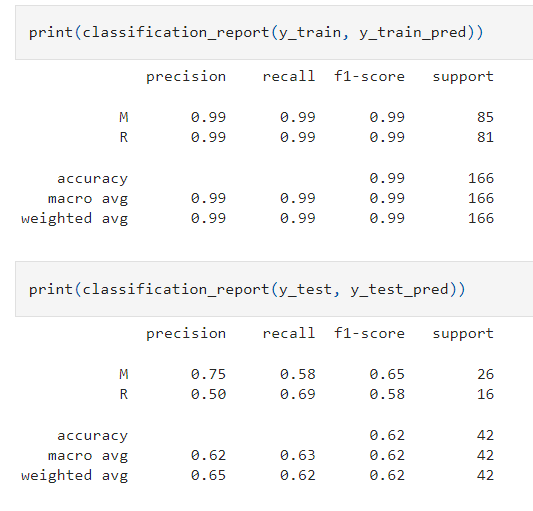
Es un algoritmo de clasificación que usa un modelo basado en un árbol de decisiones con sus consecuencias para predecir resultados.





Podemos ver con con el conjunto de train los resultados obtenidos son óptimos, sin embargo al trasladar el análisis al conjunto de test los resultados son mucho peores. Motivo por el que podemos concluir que ha habido overfitting o *sobreajuste*. Con esto en mente, realizamos Pre-Prunning usando GridSearch para encontrar el arbol más óptimo de entre una serie de árboles con unos parámetros de hojas y anchura ya dados:

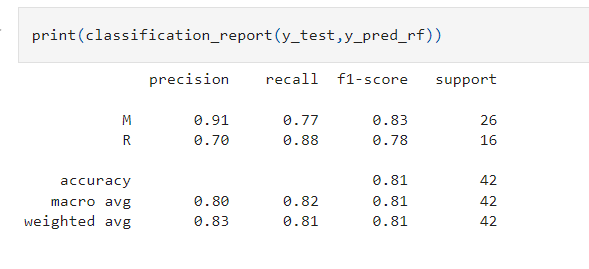




Habiendo realizado el Pre-Prunning vemos cómo ya no se produce overfitting como en el caso anterior pero los resultados obtenidos siguen lejos de ser óptimos con una precisión de 0,75.

## **RandomForest**

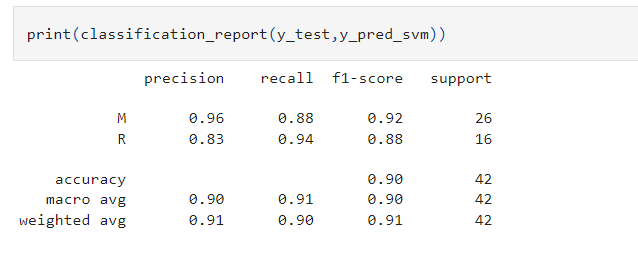
Es un algoritmo utilizado para clasificación basado en una combinación de árboles predictores tal que cada árbol depende de los valores de un vector aleatorio probado independientemente y con la misma distribución para cada uno de estos.

****

Para RandomForest, el valor de **precisión** buscado tiene un valor de 0.91

## **SVM**

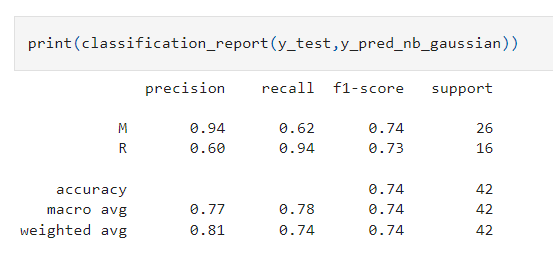
El objetivo de SVM es encontrar un hiperplano que separe dos clases diferentes de puntos de datos con el margen más amplio entre ambas clases.

****

La precisión para SVM es de 0.96, siendo esta la mayor hasta el momento.

## **Naive-Bayes**

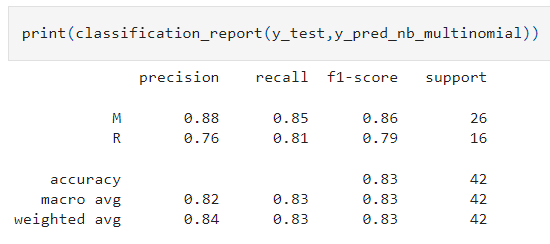
Es un algoritmo basado en el teorema de bayes para clasificar objetos que asume que las variables del dataset no están relacionadas entre ellas.

****

Para Naive-Bayes tiene un 0.94 de precisión.

## **Naive-Bayes Multinomial**

Se trata de un algoritmo basado en el teorema de Bayes para la clasificación de objetos que asume la independencia entre las variables del dataset, se diferencia de Naive-Bayes en que cuenta con un vector que indica el número de apariciones de cada elemento en particular.

****

La precisión para Naive-Bayes Multinomial es de 0.88 de precisión.

# **Modelo elegido**

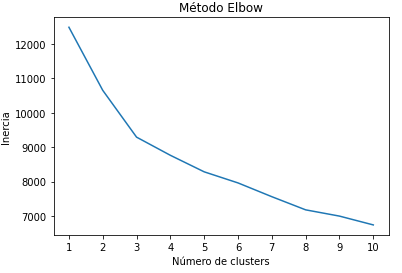
Por lo mostrado en el apartado anterior decidimos que el mejor modelo para este problema es SVM con un 0.96 de precisión.

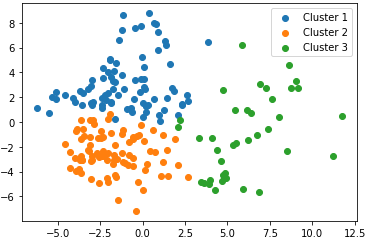
# **Clustering**

Hemos utilizado algoritmos de aprendizaje no supervisado para intentar encontrar características comunes que permitan distinguir entre los metales y las rocas.

## **Elección del número de clusters**

Utilizando el método del codo sobre el dataset, como el punto en el que se observa el cambio más brusco en la inercia nos dirá el número óptimo de Clusters, y en este caso ese punto es 3, el número total de clusters escogidos por el método del codo es 3.



****

No obstante, no parece muy clara la distinción entre clusters, especialmente en la frontera de los Clusters 1 y 2. Por ello, vamos a buscar el número óptimo de clusters usando otro método.

Para ello vamos a utilizar la métrica de silhoutte la cual consiste en medir cómo de similar es un objeto de un cluster con sus vecinos pertenecientes al mismo y la diferencia con objetos de otros clusters, este ofrece valores comprendidos entre -1 y 1, donde un valor alto indica que un objeto se relaciona de forma adecuada con sus vecinos y se diferencia claramente de elementos de otros clusters. Con esta métrica los resultados obtenidos son:

Para k = 2 La media de silhouette es : 0.2178194633782484

Para k = 3 La media de silhouette es : 0.13471787253074766

Para k = 4 La media de silhouette es : 0.13649098052831896

Para k = 5 La media de silhouette es : 0.12229111527357212

Para k = 6 La media de silhouette es : 0.13686609352797682

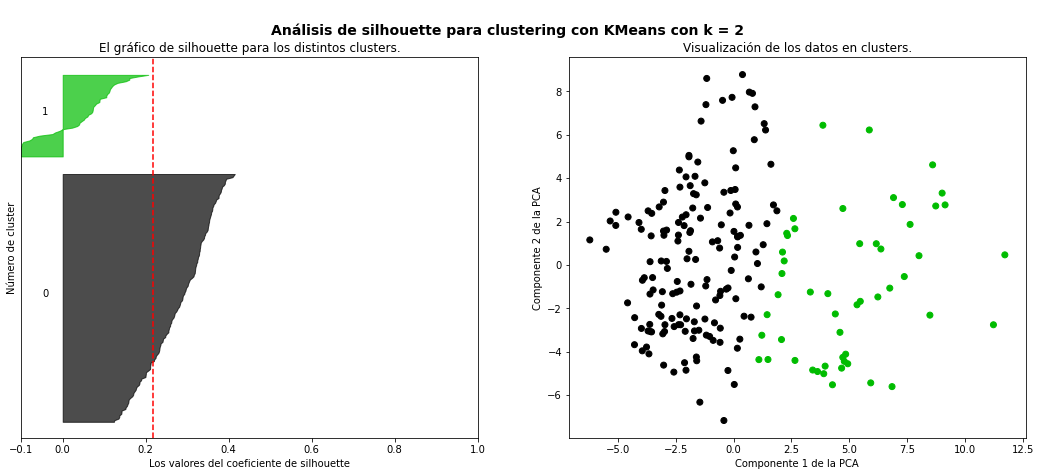
Para k = 7 La media de silhouette es : 0.10803864268778789

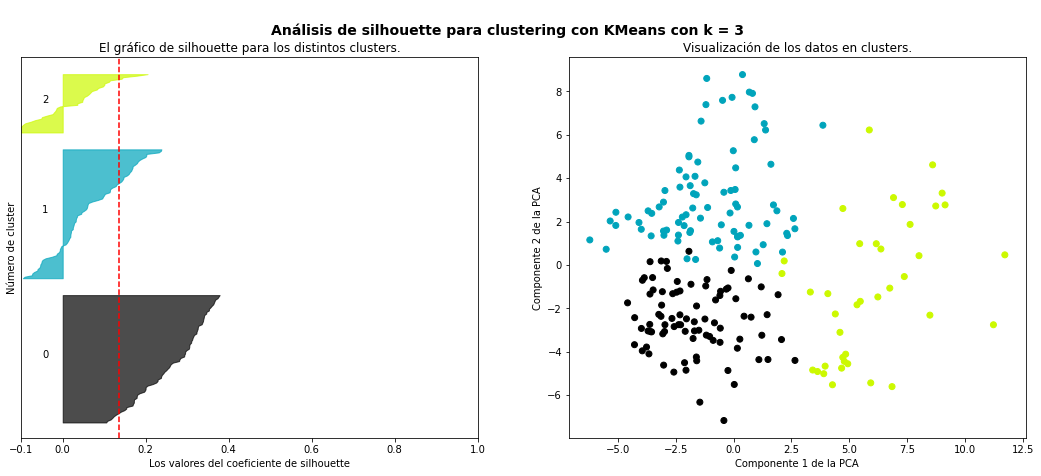
Para k = 8 La media de silhouette es : 0.11879259314239257

Para k = 9 La media de silhouette es : 0.1165921836401728

Para k = 10 La media de silhouette es : 0.11484793099580967

Como la media más alta de los valores es 0.2178… para k = 2, el número recomendado de clusters es 2.

****

****

Viendo la gráfica para ambos números de clusters (k = 2 y k = 3), podemos ver que para k = 2 sí hay una distinción algo más marcada entre los miembros de uno y otro conjunto.

**Conclusión de los resultados**

Con los resultados obtenidos, podemos asegurar que de cada 100 cilindros que clasifiquemos como un metal, 96 serán siempre metal. Esto garantiza el menor número posible de falsos positivos y la mayor seguridad para la atracción.

Por otro lado, para hacer una mejor interpretación de los clusters, se requeriría de trabajo adicional. Una vez visto que bajo nuestro criterio el número más óptimo de conjuntos es de 2 (lo cual coincide con el número de categorías para nuestro problema de clasificación), Podríamos añadir el cluster como variable en el conjunto de datos para ver si los 2 conjuntos obtenidos se corresponden con la categoría de metal y roca o hacer otros estudios.

# 