Logotipo

Descripción generada automáticamenteInforme de practica de Aprendizaje computacional:

Regresión lineal

Luis Fernando Paz Galeano 1567369

Miguel Del Arco Márquez 1566698

Contenido

[Introducción 3](#_Toc86250693)

[Objetivos 4](#_Toc86250694)

[Apartado C: Análisis de los datos 5](#_Toc86250695)

[Información sobre el dataset 5](#_Toc86250696)

[Carga del Dataset 5](#_Toc86250697)

[Atributos del dataset 5](#_Toc86250698)

[Valores nulos 7](#_Toc86250699)

[Estadística de los datos 9](#_Toc86250700)

[Visualización de los ‘n’ elementos 9](#_Toc86250701)

[Resolución de las preguntas 14](#_Toc86250702)

[Apartado B: Regresión lineal 15](#_Toc86250703)

[Regresión lineal simple 15](#_Toc86250704)

[Error cuadrático medio y r2 17](#_Toc86250705)

[Regresión lineal simple con Normalización 18](#_Toc86250706)

[PCA 21](#_Toc86250707)

[Componentes principales 21](#_Toc86250708)

[Resolución de las preguntas 24](#_Toc86250709)

[Apartado A: Gradient descent 25](#_Toc86250710)

[Regresión polinomial aplicada 27](#_Toc86250711)

[Resolución de preguntas 28](#_Toc86250712)

[Conclusiones 29](#_Toc86250713)

# Introducción

En el presente informe se detallan los pasos seguidos para el desarrollo de la practica 1 de la asignatura de Aprendizaje computacional, basada en regresión lineal.

La práctica consiste en tres aparatados, los cuales se exponen en el informe de la siguiente manera, apartado C ´*Análisis de los datos*´, apartado B ‘*Regresión lineal*’ y apartado A ‘*Gradient descent’*.

Para el apartado C, se ha procedido a analizar la base de datos inicial, con el objetivo de poder entender la distribución, naturaleza y relación de cada uno de los atributos del conjunto de datos, para ello se han utilizado funciones propias de las diferentes librerías de análisis de datos proporcionadas por Python, entre ellas *Matplotlib,Seaborn,Pandas y NumPy.*

Para el apartado B se implementó un modelo de aprendizaje, utilizando, regresión simple y múltiple, así también se hicieron comparativas entre diferentes aplicaciones de técnicas de análisis de datos.

La regresión es una técnica de modelado estadístico que se emplea para describir una variable de respuesta continua como una función de una o varias variables predictoras. Puede ayudar a comprender y predecir el comportamiento de sistemas complejos o a analizar datos experimentales.

El aprendizaje consiste en encontrar cuáles son los mejores parámetros (coeficientes) para los datos que tengamos. Los mejores coeficientes serán los que minimicen alguna medida de error. Para la regresión lineal usaremos el *error cuadrático medio.*

Finalmente, en el apartado ‘A’ se implementó el descenso de gradiente, que es un algoritmo de optimización iterativo de primer orden para encontrar un mínimo local de una función diferenciable, el objetivo de usar dicho algoritmo es con el fin de minimizar la función de coste, y así encontrar el modelo optimo como solución al problema.

# Objetivos

* Familiarizarse con el uso de las principales librerías de Python para *Machine learning.*
* Aprender a analizar un conjunto de datos.
* Aprender a identificar la relación entre atributos de un conjunto de datos.
* Asimilar conceptos teóricos por medio de su implementación.
* Desarrollar modelos básicos de entrenamiento.
* Desarrollar modelos con regresión lineal simple y múltiple.
* Resolución de conflictos e incongruencia en el tratamiento de datos.

# Apartado C: Análisis de los datos

## Información sobre el dataset

El dataset que se nos fue asignado fue *‘[Deodorant Instant Liking Data’](https://www.kaggle.com/ramkumarr02/deodorant-instant-liking-data)*, que consiste en un dataset que contiene una serie de encuesta, sobre un conjunto de productos, en este caso desodorantes, los cuales constan de varios atributos calificativos, y donde para cada uno de los productos se recoge una repuesta por parte de un entrevistado.

En este apartado hemos analizado el contenido de los datos con los que tratamos, esto con el objetivo de tener una primera percepción sobre la composición de los diferente atributos o elementos que conforman el conjunto de datos, así también poder identificar posible irregularidades o elementos nulos, esto para organizar de mejor manera la información que disponemos.

Para ello pasamos ahora a explicar como se han analizado los datos, antes que nada, deberemos de cargar los datos, para poder trabajar con ellos, así también se muestran la función utilizada para cargar los datos.

### Carga del Dataset

El primer paso es cargar nuestro dataset, ello la hacemos con la ayuda de la función *‘load\_dataset’* , que carga el contenido de nuestra base de datos, esto lo hacemos con la librería panda y la función ‘*read\_csv’*, y luego indicamos la ruta donde se encuentra el archivo.

|  |
| --- |
| def load\_dataset(path):      dataset = pd.read\_csv(path, header=0, delimiter=',')      return dataset |

### Atributos del dataset

Una vez que hemos cargado nuestro dataset, ya estamos listos para analizar el contenido del mismo, para ello nos auxiliamos del función *‘.info()*’, la cual nos devuelve un listado de las columnas , es decir cada uno de los diferente atributos del conjunto de datos, a su vez también nos indica el tipo de dato de cada uno de los elementos, así como el espacio que este ocupa en memoria.

|  |
| --- |
| Texto  Descripción generada automáticamente  Fig. 1 Ejemplo de salida 1 |

Como podemos observar en el caso anterior disponemos de un total de 53 columnas, es decir un total de 53 atributos, de los cuales 2 son elementos de tipo flotante y 52 de tipo integer.

### Valores nulos

Ahora procederemos a limpiar aquellos atributos que nos pueden generar problemas, es decir aquellos atributos que son nulos, y por lo no nos aportan ninguna información útil, para ello lo podemos hacer uso de la siguiente instrucción que nos devuelve una lista con el número de elementos vacíos por atributo.

|  |
| --- |
| Fig.2 Lista de elementos nulos |

Para tener una mejor percepción de lo elementos nulos, creamos una gráfica, desde donde tendremos una visión más global.

Después de ello lo que procederemos a hacer es a eliminar aquellos atributos que contengan elementos nulos o bien estos vacíos, sin información que nos sea útil.

Para ello implementamos la siguiente función.

|  |
| --- |
| Def columns\_with\_na(dframe):      temp = dframe.isna().sum()      temp = temp[temp >0]      print(f”Columns containing nan values:{temp.index}”)      return temp.index |

Esta función nos devuelve una lista con el índice de aquella columnas que contienen elementos vacíos, luego con la ayuda de la función ‘*.drop()’*, pasamos la lista de índices para eliminarlos del dataset.

|  |
| --- |
| Fig. 3 Visualización de elementos nulos 1 |

|  |
| --- |
| Imagen que contiene Gráfico  Descripción generada automáticamente  Fig. 4 Elementos no nulos 1 |

### Estadística de los datos

Podemos ver fácilmente una estadística de los datos para ellos simplemente hacemos un *‘dataset.describe()’* esto puede ser útil para analizar el comportamiento de los datos.

|  |
| --- |
| Fig. 5 Estadística de los datos |

### Visualización de los ‘n’ elementos

|  |
| --- |
| Fig. 6 Muestra de n filas 1 |

Correlación de los datos: Matriz de confusión

La matriz de confusión es una herramienta muy útil para valorar cómo de bueno es un modelo clasificación basado en aprendizaje automático. En particular, sirve para mostrar de forma explícita cuándo una clase es confundida con otra, lo cual nos, permite trabajar de forma separada con distintos tipos de error.

Cada columna de la matriz representa el número de predicciones de cada clase, mientras que cada fila representa a las instancias en la clase real. Uno de los beneficios de las matrices de confusión es que facilitan ver si el sistema está confundiendo dos clases.

En nuestro caso obtenemos la siguiente matriz de confusión, por temas de simplicidad mostramos solo los primeros 20 atributos de los 53 disponibles, ya que, si bien es necesario comprobar cada una de las posibles correlaciones, por la naturaleza de los datos de nuestro dataset, la correlación existente entre cada uno de los elementos es muy baja.

|  |
| --- |
| Fig 7 Matriz de confusión |

Como podemos observar en la matriz de confusión anterior, la correlación entre la mayoría de los atributos es muy baja, a excepción de dos atributos que son *‘Instant\_Liking’* y *‘q1\_1.personal.opinion.of.this.Deodorant’.*

Como vemos existe una correlación negativa entre ambos atributos, por lo cual hemos decidido utilizarlos como datos de entrada y salida de nuestro modelo de regresión lineal.

Hay que mencionar que los datos contenidos en *‘Instant\_Liking’* son categóricos, es decir, guardan una repuesta de Si o No, representado por 0 y 1.

#### Representación de Instant Liking en base a producto

|  |
| --- |
| Fig. 8 Instant Liking |

*Numero de Entrevistados por cada producto*

|  |
| --- |
| Fig. 9 Entrevistado por producto |

#### Representación de media de respuestas por producto.

Hemos calculado la media de las respuestas de cada uno de los atributos, si bien este valor no resulta determinante para determinar o entrenar algún modelo, a nivel conceptual nos ayuda a hacernos una idea de las repuestas medias de cada uno de los productos, y así determinar, por ejemplo, cuál de los productos tiene una cualificación mayor, que como recordemos, no tiene que ser precisamente el producto mejor valorado, ya que los atributos son muy variados.

|  |
| --- |
| Fig. 10 Media de respuesta por productos |

*Histograma de todos los elementos*

Podemos visualizar el rango de cada uno de los atributos, estos nos ayudan a entender mejor el comportamiento de los datos del dataset.

|  |
| --- |
| Fig. 11 Histograma de cada atributo |

## Resolución de las preguntas

**¿Cuál es el tipo de cada atributo?**

Para determinar el tipo de atributo, podemos hacer uso de la función dataset.info () que nos muestra el tipo de valor de cada uno de los elementos de los que está compuesto nuestro dataset.

Por ejemplo, en nuestro caso, al ejecutar dicho comando nos devuelve una lista de cada una de las columnas, y al final del todo nos proporciona la siguiente información:

dtypes: float64(10), int64(53), object(1)

**¿Qué atributos tienen una distribución Gaussiana?**

En nuestro caso para poder determinar que atributo tenía dicha distribución, hemos hecho uso de la comanda dataset.hist(figsize= (20,20)), esto nos devuelve un histograma para cada uno de los atributos del dataset, y hemos identificado que el atributo q7 y la media tienen una distribución Gaussiana.

**¿Cuál es el atributo objetivo? ¿Por qué?**

Después del análisis de los datos que disponemos, llegamos a la conclusión que lo ideal sería predecir el tipo de producto en base al conjunto de 'Xs' de entrada, ya que en este caso disponemos de un sin número de entradas con valores calificativos de diferentes características del producto.

# Apartado B: Regresión lineal

## Regresión lineal simple

En el apartado C, después de analizar los datos, nos dimos cuenta de una correlación negativa entre los atributos ‘Instant\_Liking’ y ‘q1\_1.personal.opinion.of.this.Deodorant', por lo cual hemos entrenado un modelo de regresión lineal simple, teniendo como entrada ‘Instant\_Liking’ y elemento de salida ‘q1\_1.personal.opinion.of.this.Deodorant'.

Queremos mencionar que, por la relación y distribución de los datos, para este dataset, la regresión lineal, no es la mas apropiada, ya que lo ideal sería implementar una clasificación en su lugar, por la manera en que está planteada la base de datos, es decir contamos con un conjunto de valores que son categóricos, calificativos y que en su mayoría son subjetivos.

|  |
| --- |
| Fig. 12 Relacion de Instant\_Liking y personal.opinion.of.this.Deodorant' |

Hemos separado los datos en entramiento y prueba, decidimos designar un 20% para la parte de pruebas, así también se definió el tipo de algoritmo como *linearRegresion.*

|  |
| --- |
| X\_train,X\_test,y\_train,y\_test = train\_test\_split(X\_train,y\_train,test\_size=0.2,random\_state=15)  #definimos el algoritmo a utilizar  lr = linear\_model.LinearRegression()  #Entreno de modelo  lr.fit(X\_train, y\_train)  #Realizar una prediccion  Y\_pred = lr.predict(X\_test) |

El resultado obtenido es el siguiente:

|  |
| --- |
| Fig. 13 Regresion Lineal Simple |

### Error cuadrático medio y r2

El coeficiente R2 determina la capacidad de un modelo para predecir futuros resultados. El mejor resultado posible es 1.0, y ocurre cuando la predicción coincide con los valores de la variable objetivo.

Por otro lado, el Error cuadrático medio es el estimador que mide el promedio de los errores al cuadrado, es decir, la diferencia entre el estimador y lo que se estima.

Las mediciones sobre el Error cuadrático medio y el porcentaje de r2 obtenidos fueron los siguientes.

|  |
| --- |
| **Mean squeared error:** 27.188  **R2 Score:** 0.6636267790895974 |

Segundo Ejemplo de regresión lineal simple

Para poder ver el contraste con la regresión lineal anterior entrenamos otro modelo, solo que esta vez con un valor de salida diferente ‘q9.how.likely.would.you.be.to.purchase.this.Deodorant’, y este valor en sí, contiene una correlación muy baja respecto al valor de entrada original.

Los resultados obtenidos fueron los siguientes:

|  |
| --- |
| Fig. 14 Regresión Lineal Segundo Ejemplo |

|  |
| --- |
| Fig.15 Regresión lineal Ejemplo 2 |

Las mediciones sobre el Error cuadrático medio y el porcentaje de r2 obtenidos fueron los siguientes.

|  |
| --- |
| Mean squeared error: 13.199  R2 Score: 0.0012714162780590454 |

## Regresión lineal simple con Normalización

Por regla general realizar una normalización ayuda a tener una mejor predicción y como actualmente no tenemos una buena predicción decidimos realizarla.

Pero antes de empezar vamos a limpiar nuestros datos ya que tenemos atributos que no podemos normalizar porque son cadenas de texto o no tienen sentido normalizar como un id, por ejemplo.

|  |
| --- |
| to\_drop=['Respondent.ID','Product.ID','Product']  dataset\_copy.drop(to\_drop, axis=1, inplace = True) |

Para la normalización existen varios métodos para realizarla como por la media, min-max, cuantiles etc.

En nuestro caso hemos usado la de la media ya que es la mas que realizar y que los puntos estén bien distribuidos.

|  |
| --- |
| dataset\_copy=dataset\_copy.apply(lambda x: (x-x.mean())/ x.std(), axis=0)  dataset\_norm=(dataset\_copy - dataset\_copy.min()) / ( dataset\_copy.max() - dataset\_copy.min()) |

Si vemos un ejemplo de cómo queda podemos observar que los valores están entre un rango del 0 al 1

|  |
| --- |
| Pantalla de computadora con números  Descripción generada automáticamente con confianza media  Dataset normalizado |

Definimos los datos de entrenamiento

|  |
| --- |
| data = dataset\_norm.values  X\_train=data[:,np.newaxis,3]  #Defino los datos correspondientes a las opinion de desodorante  y\_train=dataset['q1\_1.personal.opinion.of.this.Deodorant'] |

|  |
| --- |
| Gráfico, Gráfico de dispersión  Descripción generada automáticamente |

Regresión lineal Múltiple

En el caso de la regresión múltiple, sigue la misma idea que la distribución lineal simple, solo que, en este caso, en lugar de tener solo un elemento de entrada.

Hemos eliminado con anterioridad aquellos elementos que no consideremos relevantes, como puede ser el nombre del producto o el identificador del entrevistado, identificador del producto, para dejar solo aquellos valores cuantitativos.

La intención aquí es ver si al aumentar el numero de valores de entrada, mejora la predicción del atributo de salida.

|  |
| --- |
| def regresion\_multiple(X\_multiple,y\_multiple):      #Separamos los datos de "train" en entrenamiento y prueba      X\_train,X\_test,y\_train,y\_test = train\_test\_split(X\_multiple,y\_multiple, test\_size=0.2)      #definimos el algoritmo a utilizar      lr\_multiple = linear\_model.LinearRegression()      #Entreno de modelo      lr\_multiple.fit(X\_train,y\_train)      #Realizar una prediccion      Y\_pred\_multiple = lr\_multiple.predict(X\_test)      print('Datos del modelo de regresion multiple')      print('--------------------------------------')      print('Valor de las pendientes o coeficiente "a":')      print(lr\_multiple.coef\_)      print()      print('Valor de las interseccion de  "b":')      print(lr\_multiple.intercept\_)      print()      print('Precision del algoritmo:')      print(lr\_multiple.score(X\_train,y\_train)) |

Los resultados obtenidos, teniendo como entrada todos los atributos del dataset y como salida ‘q1\_1.personal.opinion.of.this.Deodorant’.

|  |
| --- |
| Datos del modelo de regresión múltiple -------------------------------------- Valor de las pendientes o coeficiente "a": [-2.87232144e+00 -1.28503551e-02 1.75375252e-02 -3.49271562e-03 -2.90928830e-03 -1.42650151e-02 -1.93429145e-02 1.20667624e-02 4.76773398e-03 1.09342594e-02 -1.11274365e-02 -8.33170090e-03 4.61099765e-03 -3.78023751e-02 2.63231820e-03 -3.07110586e-03 6.67705641e-03 -3.50230345e-02 -8.57705884e-03 1.33047962e-02 -3.05662017e-03 -8.92295377e-03 2.73982188e-04 -4.25617068e-02 -3.42994981e-03 -1.48858778e-03 -3.10020789e-02 -1.85378770e-02 -4.02585896e-02 9.91008920e-02 6.74074689e-02 -3.97588019e-02 4.06046968e-02 6.36581068e-02 2.55301145e-01 2.60624347e-03 -5.79583686e-03 -2.64231458e-02 9.00376535e-03 2.00212899e-03 -6.83381756e-03 1.71021217e-02 -5.20417043e-18 -2.64908827e-02 -5.46966807e-03 -1.08482735e-02 -8.82712675e-03 -4.46779841e-03 2.85423281e-01 -2.70940905e-02 -1.22112708e-03 1.56617846e-02]  Valor de las intersecciones de "b": 6.547331377775951  Precisión del algoritmo**: 0.6790243866496062** |

Como podemos observar la mejora es insignificante, esto se debe en parte ya que los demás atributos que forman parte de dataset, no son determinantes para predecir el atributo de salida que deseamos, esto ya era predecible, ya que como vimos en la matriz de confusión, la correlación de los datos era muy baja.

A manera de contraste, procedimos a realizar una predicción múltiple, solo que esta vez sin tener en cuenta el atributo ‘Instant\_Liking’, que como sabemos tiene una correlación negativa significativa con el atributo de salida que deseamos predecir.

|  |
| --- |
| Datos del modelo de regresión múltiple -------------------------------------- Valor de las pendientes o coeficiente "a": [ 1.65689267e-02 -1.60120190e-02 1.07609274e-02 -8.92367491e-03 -8.90682181e-03 1.38127379e-03 -2.92575722e-02 7.91417059e-03 -7.00644525e-03 -3.71164299e-02 8.50980837e-04 -6.11562441e-03 -2.60235329e-02 -7.36597423e-03 -2.34113172e-02 -1.40176226e-03 -3.33888138e-02 -4.64428335e-02 -1.98050447e-02 5.42875969e-03 -1.56774366e-02 8.63757431e-03 -4.52677814e-02 2.72515973e-02 1.12761910e-02 -2.70001839e-02 -2.85254519e-02 -3.39217099e-02 -1.24776605e-01 1.02293059e-01 2.07487614e-01 -7.73020189e-02 -5.75953995e-02 -1.22358007e+00 -3.86141497e-03 -5.39210648e-02 -1.30820439e-02 -4.41524070e-03 -1.78841830e-02 -9.95114746e-03 3.41689023e-02 -1.66533454e-16 -4.10314281e-02 2.45278746e-02 3.10216477e-03 -5.94637418e-02 -9.83717018e-03 -9.02038250e-02 -8.91133648e-02 5.72036615e-03 2.73091793e-02]  Valor de la intersección de "b": 6.474639529247118  Precisión del algoritmo**: 0.02064575037155192** |

Como vemos el porcentaje de la predicción en este caso es muy bajo, por lo cual concluimos también que el atributo más determinante es ‘Instant\_Liking’.

## PCA

### Componentes principales

Hemos calculado el número de componentes principales en base a la ratio de varianza explicada, es decir la cantidad de información que es capaz de capturar, cada uno de los componentes principales obtenidos.

|  |
| --- |
| Fig. 16 Componentes principales 1 |

Como vemos la cantidad de componentes principales de nuestro dataset deja de ser significativa a partir de los 7 elementos aproximadamente, por lo cual, en esta situación, escogemos como numero de componentes principales 4, si lo graficamos obtenemos lo siguiente:

|  |
| --- |
| Fig. 17 PCA |

|  |
| --- |
| Fig. 18 PCA |

## Resolución de las preguntas

**¿Cuáles son los atributos más importantes para realizar una buena predicción?**

En nuestro caso el atributo más importante es el Instant.Liking ya que hemos comprobado que, de entre los múltiples elementos que forman nuestro dataset, este dispone de una correlación más pronunciada con estos.

**¿Con qué atributo se alcanza un MSE menor?**

Si bien  el MSE, menor se puede obtener de varios atributos , que podríamos establecer como relación, de la regresión que queramos analizar, y siendo el MSE '0', solo en aquel caso donde los valores de salida, estén contemplados en los valores de entrada, es decir que sean los mismos, tanto los de entrada como salida, en el apartado de regresión lineal simple, hemos planteado dos ejemplos que nos parecieron prácticos para realizar el cálculo del MSE, nos dimos cuenta si bien para los atributos  de entrada seleccionados, siendo uno 'q9.how.likely.would.you.be.to.purchase.this.Deodorant' nos generaba un MSE menor, sin embargo, la predicción era muy baja.

**¿Qué correlación existe entre los atributos de su base de datos?**

Prácticamente la correlación existente es muy baja, a excepción de una correlación negativa entre Instant.Liking y q1\_1.personal.opinion.of.this.Deodorant.

**¿Cómo influye la normalización en la regresión?**

Al intentar normalizar los datos, en nuestro caso lo que sucede es que se pierde la relación entre los atributos sobre los cuales queremos realizar la regresión, ya que por una parte uno de los valores de entrada son de tipo binarios, mientras que los de salida va de un intervalo del 1 al 7, con lo cual en la regresión obtenida, se refleja una relación errónea entre los valores, ya que al normalizar, todos los valores van en intervalos del 1 al 0, con esto no consideremos que la normalización no sea necesaria,  sino más bien creemos que por la naturaleza y relación de los datos de nuestro dataset, en este caso influye negativamente.

**¿Cómo mejora la regresión cuando se filtran aquellos atributos de muestras que no contienen información?**

Curiosamente no afecta ya que nuestro atributo con mayor peso es el Instant.Liking y cambiar o eliminar otros atributos no tienen el suficiente impacto para variar la regresión lineal, ya que estos no son determinantes, de hecho, el resultado de tener en cuenta más atributos además del Instant.Liking hace que la predicción varié poco.

**Si se aplica un PCA, ¿a cuántos componentes se reduce el espacio? ¿Por qué?**

Como vemos al aplicar el PCA, el espacio se reduce a 4 componentes, esto es debido a que la información a lo largo del eje o ejes principales menos importantes se elimina, dejando solo el componente de los datos con la varianza más alta.

# Apartado A: Gradient descent

|  |
| --- |
| def GradientDescent(X,y,learning\_rate,iterations):      b = 0      m = 5      n = X.shape[0]      for \_ in range(iterations):              # As using mean squared error.              b\_gradient = -2 \* np.sum(y - (m\*X + b))/n              m\_gradient = -2 \* np.sum(X\*(y - (m\*X + b)))/n                # Updating the previous values              b = b - (learning\_rate \* b\_gradient)              m = m - (learning\_rate \* m\_gradient)      return m\*X + b,b,m |

Los resultados obtenidos , teniendo como entrada un único parámetro, fueron los siguientes :

|  |
| --- |
| Fig. 19 Descent Gradient |

Si variamos los valores de aprendizaje y numero de iteraciones, obtenemos lo siguiente:

|  |
| --- |
| Fig.20 Descent Gradient 2 |

## Regresión polinomial aplicada

Hemos aplicado un algoritmo de regresión polinomial con los mismos datos de entrada y salida que hemos intentado predecir con los demás algoritmos, nuestra sorpresa fue que obtenemos el mismo resultado, con una pequeña variación en el porcentaje de predicción.

|  |
| --- |
| Fig. 20 Regresión Polinomial |

|  |
| --- |
| Precisión del modelo:  0.6613839043074066 |

Resolución de preguntas

**¿Cómo influyen todos los parámetros en el proceso de descenso? ¿Qué valores de learning rate convergen más rápido en la solución óptima? ¿Cómo influye la inicialización del modelo en el resultado final?**

Tenemos los parámetros X, learning rate y el número de iteraciones

- El valor de x nos indica desde donde empezara el algoritmo por lo general suele ser un numero aleatorio

- El valor de learning rate indica que tal grande será el siguiente paso para llegar a un mínimo

- Números de iteraciones serán las iteraciones que realizara el descenso de gradiente.

Este valor no debe de ser ni muy grande ni muy pequeño ya que si es un valor pequeño el descenso de gradiente tardara mucho en llegar a una solución optima, pero si es un valor muy grande el descenso de gradiente oscilará alrededor de los mínimos sin llegar a una solución, entonces volviendo a la pregunta inicial el learning rate debería de ir variando según la situación, pero un valor óptimo para encontrar una solución podría ser un learning rate de 0.02 aproximadamente,

**¿Qué diferencia (cuantitativa y cualitativa) hay entre su regresor y el de la librería?**

A nivel cuantitativa y cualitativo es mejor de la librería ya que esta esta optimizada al máximo, pero al final el resultado que se obtiene es el mismo ya que en el fondo el descenso de gradiente es una formula y es la que hemos replicada nosotros

**¿Ayuda la visualización a identificar aquellas muestras para las que el regresor obtiene los peores resultados de predicción?**

Si, ya que de esta forma podemos compararlo con el mejor caso y ver sus diferencias.

# Conclusiones

Durante la realización de la práctica, hemos podido aplicar alguno de los conceptos aprendidos en la teoría, además ha servido como puerta de entrada al análisis y procesamiento de datos desde un punto de vista diferente, en este caso con la aplicación de machine learning.

Si bien los algoritmos utilizados, y así como el análisis de los datos eran sencillos, hemos podido familiarizarnos con las principales herramientas y recursos para la modelización de datos.

En esta práctica con este dataset en concreto, nos hemos preguntado si podíamos predecir cual es el desodorante mejor calificado por parte de los encuestados , para ello tuvimos que realizar un análisis exhaustivo de los datos, junto a la implementación de los modelos de predicción (regresión lineal, regresión lineal múltiple y regresión polinomial) con unos resultados de predicción no muy elevados, esto se debe a que nuestro dataset no es el ideal para realizar predicciones, sino más bien los datos serian ideales para hacer una clasificación de clases, por falta de tiempo no hemos podido mostrar un ejemplo de clasificador y poder exprimir los datos al máximo.

Por otra parte, por la naturaleza, distribución y relación del dataset que se nos fue asignado, si bien las visualizaciones de las gráficas, del resultado de los algoritmos, no eran notables, o bien las predicciones no eran muy altas, esto no quería decir que los algoritmos implementados estuviesen mal, sino mas bien que no eran los ideales para aplicar por el tipo de conjunto de datos del que disponíamos.

Volviendo a la pregunta inicial, podemos asegurar que nuestros modelos de predicción no son muy fiables, el tipo de datos de este conjunto.