APC: PRACTICA 1

Regressions

Ivan Peñarando Martínez - 1599156
Joel Marco Quiroga Poma - 1504249
Ferran Martínez Reyes - 1565491

12 d'Octubre de 2022

GPA604-1530

GITHUB: [teolicht505/APC-Prac1 (github.com)]

INDEX

INTRODUCCIÓ

El primer pas per començar, és descarregar la nostra base de dades, en el nostre cas tenim una base de dades d'automòbils "Automobile.csv" que conté 26 atributs. Comencem llegint la base de dades amb pandas, trobem que la dimensionalitat és (201, 26), és a dir, tenim 201 entrades.

Primer de tot, volem fer el preprocessament de les dades, netejar-les, i preparar-les per a fer el model. Per verificar si les dades són correctes o no, podem començar visualitzant la mitjana de totes les variables numèriques, si hi ha alguna que no encaixa amb el tipus d'atribut corresponent, voldrà dir que haurem de prendre mesures per corregir-ho.

Fent “dataset.mean()” obtenim:

```
Dimensionalitat de la BBDD: (201, 26)
symboling               0.841
normalized_losses     125.189
wheel_base             98.797
length                174.201
width                  65.889
height                 53.767
curb_weight          2555.667
engine_size           126.876
bore                    3.330
stroke                  3.262
compression_ratio      10.164
horsepower            103.264
peak_rpm             5121.393
city_mpg               25.179
highway_mpg            30.687
price               13207.129
dtype: float64
```

Analitzant-ho veiem que tot sembla correcte, això també ho hem pogut comprovar amb la comanda np.unique(), que mostra els valors únics de cada atribut (tant numèric com no numèric). D'aquesta manera podem veure si hi ha valors erronis. En el nostre cas tot sembla ser correcte, ja que no hi ha cap valor que no sigui coherent:

```
for column_name in dataset.columns:
  print("\n" + column_name.upper() + " \n")
  print(np.unique(dataset.loc[:][column_name]))
```

```
DRIVE_WHEELS

['4wd' 'fwd' 'rwd']

ENGINE_LOCATION

['front' 'rear']

WHEEL_BASE

[ 86.6  88.4  88.6  89.5  91.3  93.   93.1  93.3  93.7  94.3  94.5  95.1
  95.3  95.7  95.9  96.   96.1  96.3  96.5  96.6  96.9  97.   97.2  97.3
  98.4  98.8  99.1  99.2  99.4  99.8 100.4 101.2 102.  102.4 102.7 102.9
 103.3 103.5 104.3 104.5 104.9 105.8 106.7 107.9 108.  109.1 110.  112.
 113.  114.2 115.6 120.9]
```

Comprovem també que no hi ha cap valor NaN, i que totes les dades de números en punt flotant (float) tenen un punt (".") en comptes d'una coma (","), per tant, no caldrà canviar res d'això.

APARTAT C

Pregunta 1) Quin és el tipus de cada atribut?
Pel que fa als atributs, si volem saber el tipus que tenen cadascun podem executar la següent comanda:

```
# més info sobre la BD i els seus atributs:
dataset.info(verbose=True)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 201 entries, 0 to 200
Data columns (total 26 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   symboling            201 non-null    int64
 1   normalized_losses    201 non-null    int64
 2   make                 201 non-null    object
 3   fuel_type            201 non-null    object
 4   aspiration           201 non-null    object
 5   number_of_doors      201 non-null    object
 6   body_style           201 non-null    object
 7   drive_wheels         201 non-null    object
 8   engine_location      201 non-null    object
 9   wheel_base           201 non-null    float64
 10  length               201 non-null    float64
 11  width                201 non-null    float64
 12  height               201 non-null    float64
 13  curb_weight          201 non-null    int64
 14  engine_type          201 non-null    object
 15  number_of_cylinders  201 non-null    object
 16  engine_size          201 non-null    int64
 17  fuel_system          201 non-null    object
 18  bore                 201 non-null    float64
 19  stroke               201 non-null    float64
 20  compression_ratio    201 non-null    float64
 21  horsepower           201 non-null    int64
 22  peak_rpm             201 non-null    int64
 23  city_mpg             201 non-null    int64
 24  highway_mpg          201 non-null    int64
 25  price                201 non-null    int64
dtypes: float64(7), int64(9), object(10)
memory usage: 41.0+ KB
```

Veiem com la base de dades està formada per 9 atributs enters (int64), 7 atributs decimals (float64) i 10 atributs strings (object)

Analitzant i entenent els atributs:

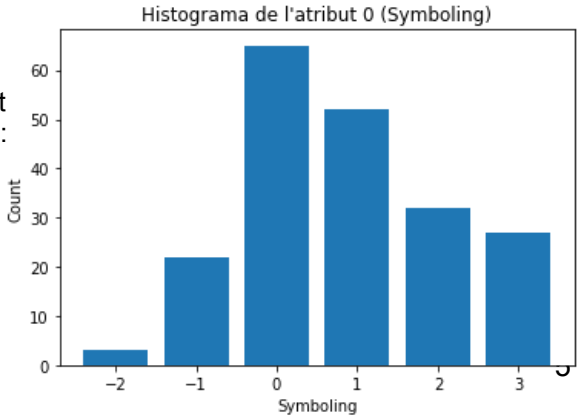
Per tal de fer l'anàlisi cal comprendre les dades amb les quals treballarem:

symboling: Enter amb el nivell de risc d'assegurança d'un cotxe (-2, -1, 0, 1, 2, 3).
normalized_losses: Float amb el pagament de pèrdua mitjana relatiu per any de vehicle assegurat.
price: Float amb el Preu de l'automobil
wheel_base: Float amb la Distancia entre eixos de les rodes
length i width: Floats amb llargada i amplada del coche, respectivament
curb_weight: Enter amb el Pes del vehicle
engine_size: Enter amb el Tamany del motor
bore: Float amb el Diametre del motor
stroke: Float amb cicle de carrera d'un motor
compression_ratio: Relació entre el volum del cilindre i la camera de combustió
horsepower: Enter amb la potencia del vehicle
peak_rpm: Enter amb la Maxima revolucio per minut
city_mpg: Enter amb la puntuació mitjana del cotxe en condicions de ciutat
highway_mpg: Enter amb la puntuació mitjana mentre condueix per una autovia a una velocitat més alta.
number_of_cylinders: String amb el numero de cilindres del motor
engine_type: String amb el Tipus de motor
make: String amb nom Marca del Fabricant
fuel_type: String amb Tipus de combustible (Gas o Diesel)
aspiration: String amb Tipus d'aspiració del motor (Standard o Turbo)
number of doors: String amb Numero de portes del cotxe ("Two" o "Four")
body_style: String amb nom de l'Estil de construcció de l'automobil
drive_wheels: String amb Tipus de transmissió de les rodes
engine_location: String amb ubicació del motor ("Rear" o "Front")
fuel_system: String amb el Tipus de sistema de combustionat

Pregunta 2) Quins atributs tenen una distribució Guassiana?

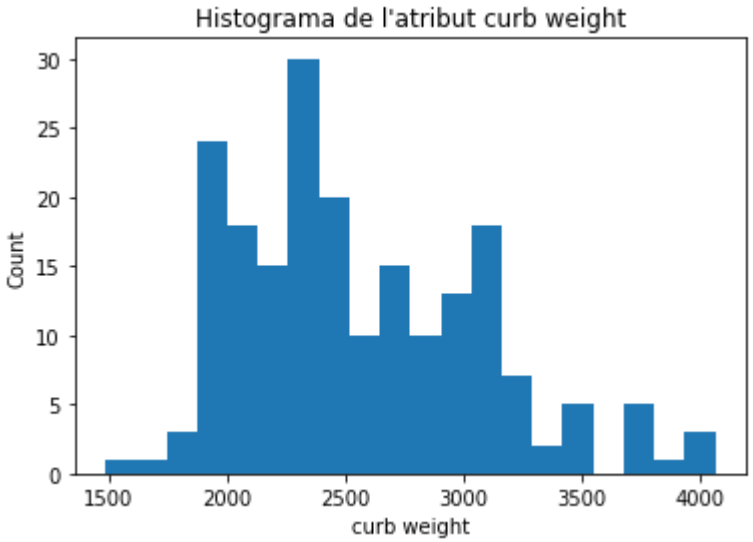
És important adondar-nos que no totes les dades tindran una distribució gaussiana, per tant, hem de descobrir quin tipus de distribució tenen. Això ho podem fer bé revisant els histogrames de les dades o implementant tests estadístics. Hem de tenir en compte, però, que les distribucions normals només s'apliquen estrictament a dades contínues, tot i que puguem utilitzar la distribució normal per aproximar dades discretes.

Per tant, atributs com "Symboling" que només poden prendre valors dins d'un conjunt de classes (-2, -1, 0, 1, 2, 3), els podem considerar com "no-gaussians" per se, tot i que sí que és veritat que podríem aproximar aquestes dades a una normal, com veiem al següent histograma:[1]



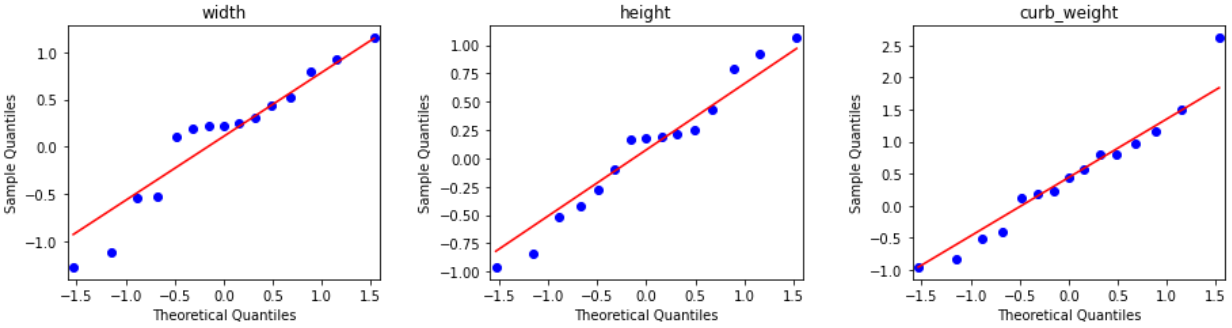
[1] Per conseguir que l'histograma es vegi correctament hem tingut que definir els bins (contenidors) seguents: [-2.5, -1.5, -0.5, 0.5, 1.5, 2.5, 3.5] Ja que symboling només pot pendre valors enters entre -2 i 3, i els bins es defineixen de la manera [X, Y) tal que X es el principi del rang del bin i Y es el final.

En canvi, atributs com "height" o "width" prenen valors continus, i, per tant, si podem parlar de la possibilitat que siguin gaussians, com veiem en el següent histograma:



Per concloure si són gaussianes o no, podríem agafar l'histograma i comparar-lo amb una corba d'una distribució de gauss, però llavors podríem passar per alt el fet que l'histograma pugui canviar en funció de l'amplada de cada bin (contenidor) i pot existir confusió en la manera d'interpretar si es tracta d'una normal o no. És per això que hem decidit fer un test de la normalitat de cada atribut a partir d'un Q-Q plot, un mètode gràfic per comparar distribucions a partir del dibuix dels seus quantils en una recta. Si comparem cada distribució amb una distribució normal, el diagrama Q-Q ens permetrà veure la desviació molt millor que en un histograma. En fer aquest dibuix dels quantils, si veiem que la majoria de punts cauen sobre la recta, podrem afirmar que es tracta d'una distribució gaussiana. Sí que és veritat que les dades quasi mai es poden considerar perfectament gaussianes, però si tenen una distribució semblant, i si la mida de la mostra és prou gran, la podrem tractar com a tal.

A continuació mostrem com a exemple una petita part del resultat obtingut:



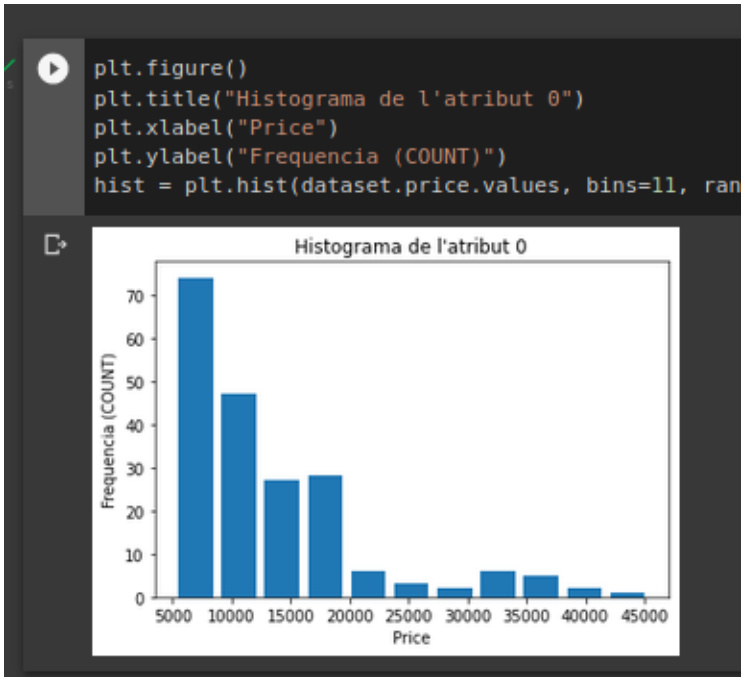
Podem veure que l'atribut width o height no segueixen la línia tan bé com ho fa curb_weight, per tant, podríem considerar curb_weight com un atribut amb una distribució gaussiana, mentre que l'atribut width o height quedarien més aviat descartats.
Un cop fet això per a tots els atributs, podem treure conclusions sobre quins segueixen distribucions normals i quins no. Segons el nostre criteri, Quedaria de la següent forma:

GAUSSIANS:
- Normalized_losses
- Wheel_base
- Curb_weight
- Engine_Size
- Horsepower
- City_mpg
- Peak_rpm

NO GAUSSIANS:
- Tots els atributs discrets, o atributs no-numerics
- Width
- Height
- Length
- Highway_mpg
- Compression_ratio
- Bore
- Price
- Stroke

Pregunta 3) Quin és l'atribut objectiu? Per què?

Hem decidit agafar com a atribut objectiu el de preu (price), ja que ens sembla el més interessant de tots en el sentit que, donades unes certes circumstàncies, si es té un vehicle i es vol saber el preu per tal de posar-ho a la venda, volem que ens predigui un preu raonable i que no se surti gaire del normal a partir de qualsevol de les dades sobre el nostre vehicle que ja tinguem. És a dir, pot ser una eina molt bona per a taxar vehicles. A més a més, es tracta d'un atribut amb suficients valors variables com per així fer-ho, tal com veiem a la següent gràfica:



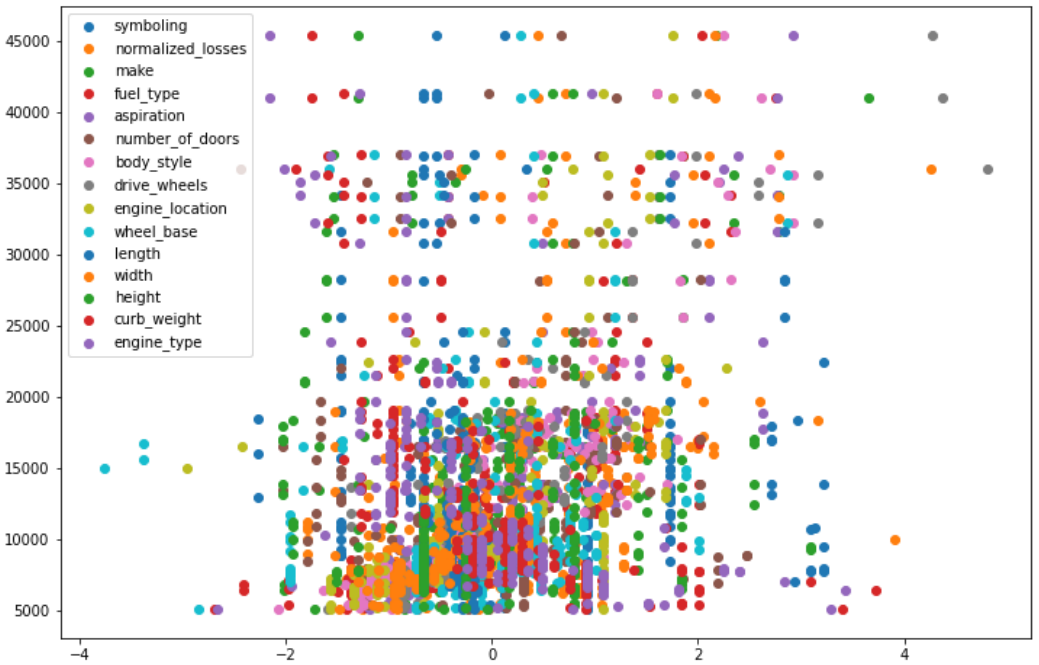
A més, escollir aquest atribut ens permet la llibertat d'expressar-lo en valors ordinals, és a dir, jerarquitzar els resultats. Per exemple un preu de 5.000 a 10.000 es podria considerar "Baix". Un preu entre 10.000 i 25.000 es podria considerar "Mitja", un entre 25.000 i 30.000 seria "Alt", mentre que un entre 30.000 i 45.000 seria un preu "Molt Alt".

APARTAT B

Normalització i Estandardització de les dades:
Abans de tot hem de tenir en compte que per a tasques de regressió és important normalitzar les dades. Això és per tal d'evitar que els atributs amb valors dins d'un rang molt gran, per exemple (100 → 1000) es tinguin més en compte (tinguin major pes en la regressió) que no pas atributs amb un rang menor, per exemple (0 → 10), ja que al calcular el MSE, els valors petits no contribuiran quasi res al càlcul del Mean Squared Error, i, per tant, els atributs amb un rang menor seran més aviat ignorats sota l'ombra d'atributs amb rangs més elevats.
Per tal de fer això, hem de normalitzar les dades, és a dir, reescalar els valors de tots els atributs perquè comparteixin un mateix rang. Per fer això, és tan senzill com utilitzar:

```
x = StandardScaler().fit_transform(x)
```

A més de normalitzar les dades, aquesta funció les estandarditza, és a dir, fa que la mitja sigui zero i que la desviació estàndard sigui 1. Podem comprovar que s'ha fet correctament fent un plot i observant que totes les dades comparteixen un rang semblant:

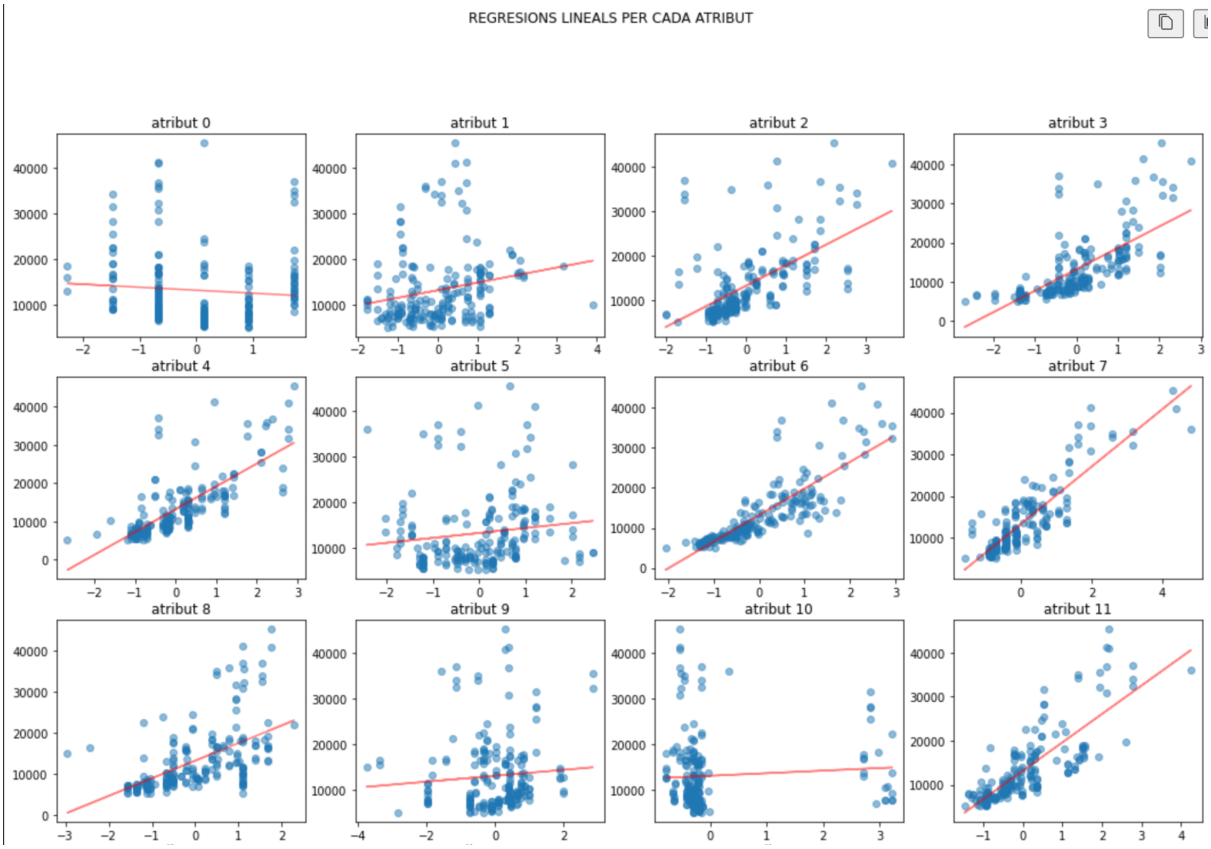


A més, si fem la mitjana de les dades un cop estandarditzades veurem com és un valor molt proper a zero:

```
[26] # Fem l'escalat estandaritzat pels atributs
     from sklearn.preprocessing import StandardScaler
     sc = StandardScaler()
     x = sc.fit_transform(x)
     print(x.mean())

     -5.8917308106144295e-18
```

Regressions per cada atribut:
hem fet les grafiques de les regressions per cada atribut i es pot veure quins atributs fan una millor predicció.



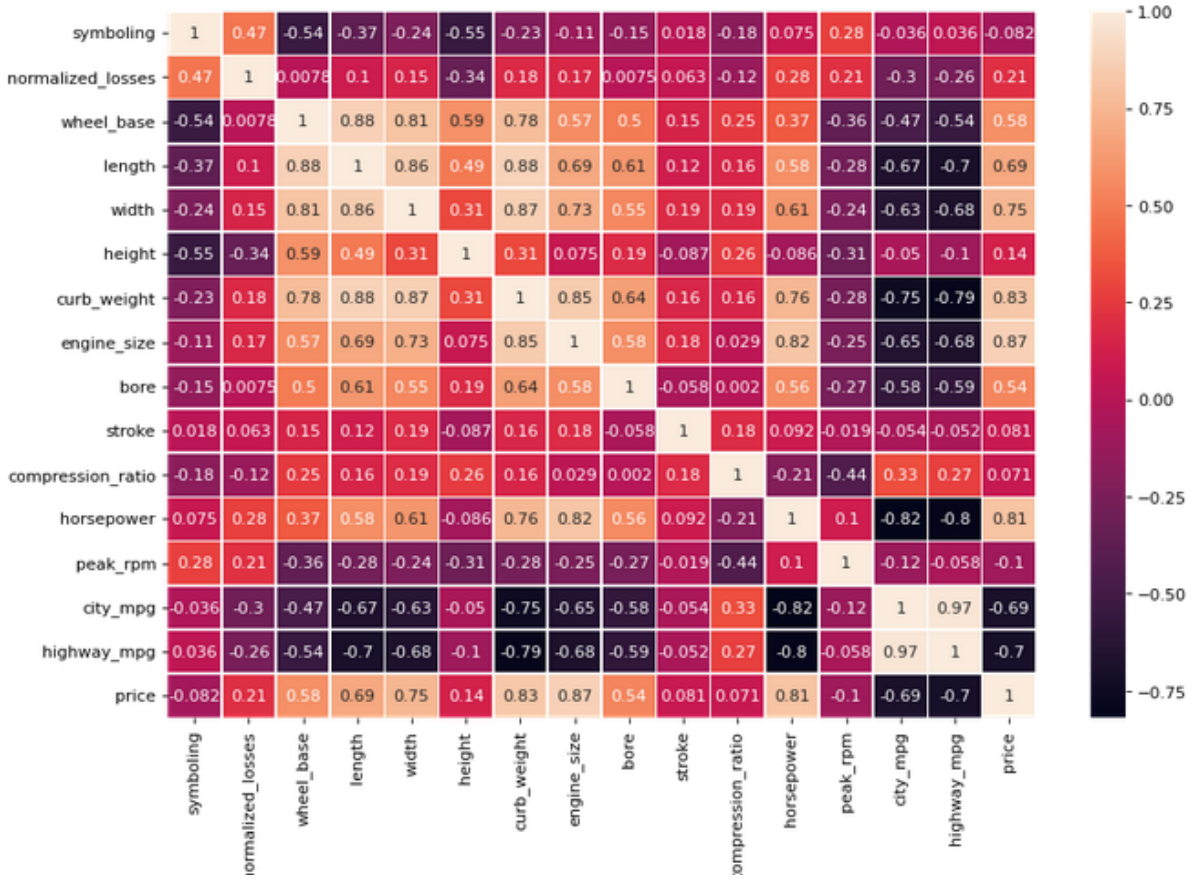
1. Quin són els atributs més importants per fer una bona predicció?

Si seleccionem molt pocs atributs, es produirà underfitting, és a dir, el model no serà el prou bo per a predir el preu. Si, en canvi, seleccionem massa atributs, es produirà overfitting, és a dir, el model serà massa semblant a l'input de la base de dades, i no serà generalitzable per a altres conjunts de dades.
Per tenir una bona predicció cal seleccionar el nombre d'atributs tal que el MSE sigui mínim. Hem de tenir en compte que si tenim variables independents (X's) amb una forta correlació entre elles, aquestes canviaran a l'unison, i en conseqüència al model li costarà estimar la relació entre les variables independents i el cost (variable dependent).

Variables que tinguin una correlació lineal amb l'outcome s'eren els atributs més importants. A més, hauríem de mirar que aquestes tinguessin suficient variabilitat i no estiguessin correlacionades amb altres variables input del model, ja que sinó crearíem un problema de col·linearitat (per exemple highway_mpg i city_mpg).

En el nostre cas, hem fet un heatmap i un pair plot de les correlacions. A partir d'això i del grau de correlació hem pogut trobar aquells atributs que tenen major correlació lineal amb el preu. Ho hem fet amb l'ajuda de la següent taula:

Scale of correlation coefficient	**Value**
$0 < r \leq 0.19$	Very Low Correlation
$0.2 \leq r \leq 0.39$	Low Correlation
$0.4 \leq r \leq 0.59$	Moderate Correlation
$0.6 \leq r \leq 0.79$	High Correlation
$0.8 \leq r \leq 1.0$	Very High Correlation



Una possible llista de candidats (i per ordre d'importancia) seríen:

- engine_size (0.87 → Very High Correlation)
- curb_weight (0.83 → Very High Correlation)
- horsepower (0.81 → Very High Correlation)
- width (0.75 → High Correlation)
- length (0.69 → High Correlation)
- wheel_base (0.58 → Moderate Correlation)
- bore (0.54 → Moderate Correlation)

Problemes de col·linearitat:

- engine_size té una alta correlació amb curb_weight i amb horsepower
- width i length tenen una alta correlació entre si
- wheel_base té una alta correlació amb width i length

Possible solucio:

Quedar-nos només amb:

- engine_size (0.87)
- width (0.75)
- bore (0.54**)**

2. Amb quin atribut s'assoleix un MSE menor?

L'atribut amb menor MSE assolit ha sigut **engine_size** tal com podem veure al resultat de l'execució:

```
MSE atribut  symboling :  62415066
R^2: atribut  symboling :  -145.31211971717488

MSE atribut  normalized_losses :  60099658
R^2: atribut  normalized_losses :  -20.918211910352618

MSE atribut  wheel_base :  41361996
R^2: atribut  wheel_base :  -0.92563566309979

MSE atribut  length :  32868224
R^2: atribut  length :  -0.09657864415361583

MSE atribut  width :  27373849
R^2: atribut  width :  0.22820574364110746

MSE atribut  height :  61688099
R^2: atribut  height :  -52.47649857461634

MSE atribut  curb_weight :  19088303
R^2: atribut  curb_weight :  0.5637293416489209

MSE atribut  engine_size :  15021126
R^2: atribut  engine_size :  0.6858854074885816

MSE atribut  bore :  44309278
R^2: atribut  bore :  -1.3909117442321501

MSE atribut  stroke :  62424920
R^2: atribut  stroke :  -148.79544458726366

MSE atribut  compression_ratio :  62523911
R^2: atribut  compression_ratio :  -195.7749562768033

MSE atribut  horsepower :  21530266
R^2: atribut  horsepower :  0.4788297526697347

MSE atribut  peak_rpm :  62150857
R^2: atribut  peak_rpm :  -88.96965534228762

MSE atribut  city_mpg :  33219371
R^2: atribut  city_mpg :  -0.12143182987148649

MSE atribut  highway_mpg :  31635042
R^2: atribut  highway_mpg :  -0.013728844351156289
```

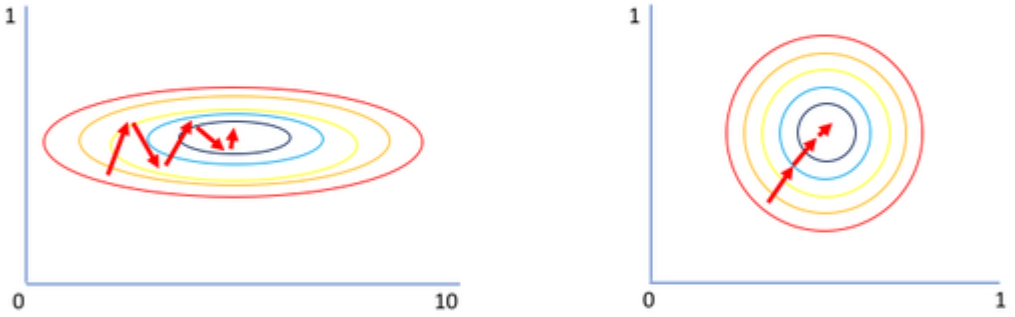
3. Quina correlació hi ha entre els atributs de la vostra base de dades?

Com ja hem vist en l'apartat B.1, les correlacions les hem obtingut tant amb un heatmap com amb un pairplot. Les següents correlacions són les més fortes:

- wheel_base amb { length, width, curb_weight }
- length amb { wheel_base, width, curb_weight }
- width amb { wheel_base, length, width, engine_size, price }
- curb_weight amb { wheel_base, length, width, engine_size, horsepower, price }
- engine_size amb { width, curb_weight, horsepower, price }
- horsepower amb { curb_weight, engine_size, price }
- price amb { width, curb_weight, engine_size, horsepower }

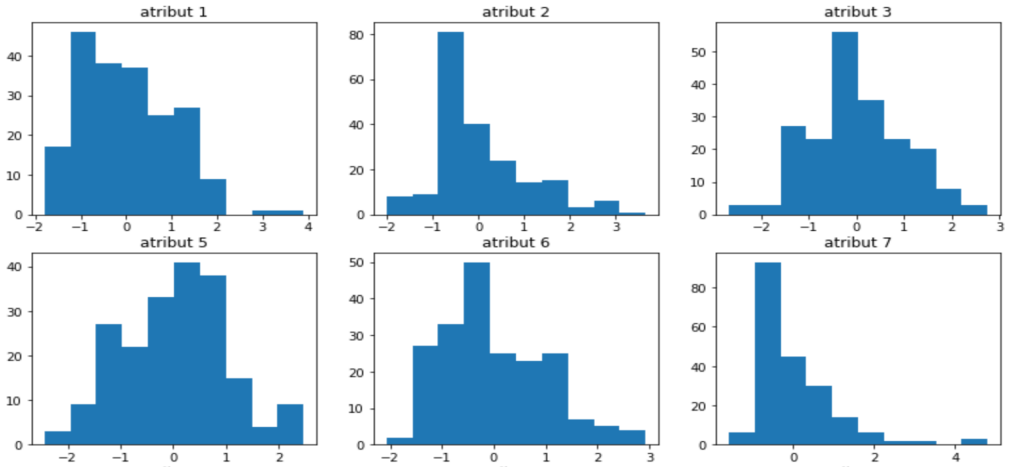
4. Com influeix la normalització en la regressió?

El fet de normalitzar les dades evitarà que els paràmetres amb uns rangs més grans dominin als de rangs més petits a l'hora de fer el descens de gradient. Com que la normalització de les dades fa que tots els paràmetres estiguin sota un mateix rang, el marge d'error (representat amb una el·lipse a l'esquerra del gràfic) disminuirà fins a convertir-se en un marge més petit (similar al cercle del gràfic). Per tant, com que el marge d'error es redueix, el descens de gradient ho tindrà més fàcil per arribar al centre (mínim del MSE) i en conseqüència el descens de gradient s'acaba fent més ràpid.



5. Com millora la regressió quan es filtren aquells atributs de les mostres que no contenen informació?

Veient les distribucions dels atributs ens adonem que totes tenen una distribució gaussiana.



Per tant, segons el criteri de trobar distribucions no gaussianes, no descartarem cap atribut. Pero ens hem adonat que tenim atributs amb els valors de RSE fora del rang -1, 1, aixo ens indica que aquest atributs nomes fan soroll per tant cal esborrar-los.

```
MSE atribut  width :  27373849
R^2: atribut  width :  0.2282057436411078

MSE atribut  height :  61688099
R^2: atribut  height :  -52.47649857461637

MSE atribut  curb_weight :  19088303
R^2: atribut  curb_weight :  0.5637293416489209

MSE atribut  engine_size :  15021126
R^2: atribut  engine_size :  0.6858854074885812
```

Per comprovar la millora hem fet ús de un mean() dels RSE i R2 de totes les rectes de regressió per tots els atributs.

Sense eliminar atributs soroll.

```
media error:  42371307.656414896
media r2:  0.265094211672302
```

Despres de eliminar atributs soroll.

```
media error:  82656812.92281465
media r2:  -0.4336345423717298
```

Podem interpretat que malgrat el MSE es pitjor al segon cas, aixo no significa que el regresor no sigui bo, nomes que hi ha mes variança al model, en canvi, podem veure que el valor de R2 ha millorat, i aixo ens indica que hi ha un millor ajustament lineal.

6. Si s'aplica un PCA, a quants components es redueix l'espai? Per què

L'estratègia del PCA s'utilitza quan hi han molts atributs a les x i es vol reduir per tal de poder representar els mateixos atributs pero en un espai més petit. L'ordre dels components ve donat per la quantitat de variància original que descriuen. És per això que es tracta d'una técnica que, tot i que reduint l'espai perdem informació, es perd la mínima possible.
En el moment d'aplicar el PCA necesitem assignar-li un número de components i és per això que hem escollit 2 components per tal de ser fàcilment visualitzable si es dona el cas.
En aquest cas el que hem fet ha sigut comparar l'error MSE y l'atribut R^2 quan les dades estan normalitzades contra les que no ho están, aconseguint el següent resultat:

```
SIN ESTANDARIZAR
Error en atributs NO estandaritzat 16_273_888.68965068
R2 score en atributs NO estandaritzat 0.6113073283160395

ESTANDARIZARIZADO
Error en atributs estandaritzats 14_546_650.7430426
R2 score en atributs estandaritzats 0.6525614345044373
```

Com es pot veure l'error es menor quan les dades estan estandaritzades i a més la puntuació r^2 és millor.

APARTAT A

Descens del gradient:

El procés de descens es basa en trobar el mínim cost donades diferents w amb l'objectiu de poder trobar un polinomi capaç de fer prediccions a partir d'unes dades que tenim d'avantmà. El polinomi que es trobarà serà de grau n, es a dir, tindrà la següent forma:

$$f(x) = w_0 + w_1 x^2 + \ldots + w_{n-1} x^n$$

Però aquesta fórmula no serà la funció de costos, ja que aquesta és solament la que ens permetrà predir d'atribut objectiu. La funció de costos és aquella que utilitzarem per actualitzar els valors de les wn, que és justament aquesta:

$$J(w) = \frac{1}{2m}\left[\sum_{i=1}^{m} (f(x^i; w) - y^i)^2 + \lambda \sum_{j=1}^{n} (w_j^2)\right]$$

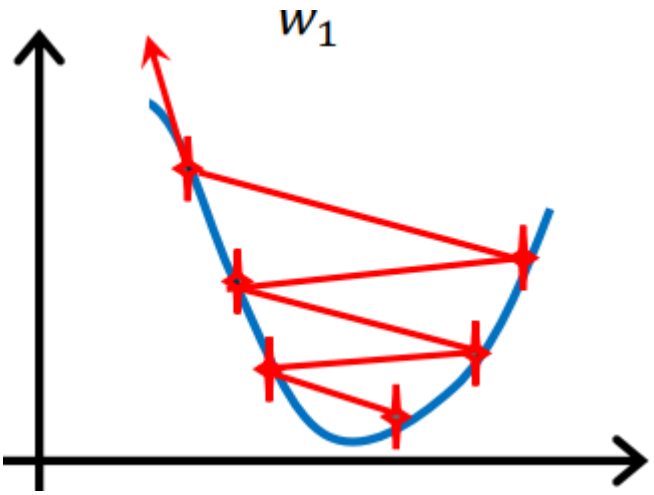
Cada vegada que calculem el cost actualitzem les w tenint en compte la següent fórmula:

$$w_j = w_j\left(1 - \alpha\frac{\lambda}{m}\right) - \alpha\frac{\lambda}{m}\sum_{i=1}^{m} (f(x^i; w) - y^i) \cdot x_j^i$$

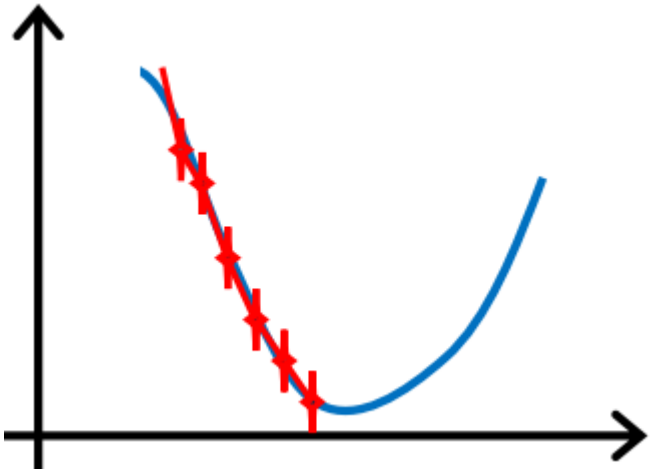
En el moment en el qual el cost J sigui superior a l'anterior (la funció divergeix) o la diferencia del cost sigui molt petita (la funció convergeix) haurem trobat un mínim que és justament el que es busca en el descens. Els diferents paràmetres del descens i la seva influència és la següent:

1. Com influeixen tots els paràmetres en el procés de descens? Quins valors de learning rate convergeixen més ràpid a la solució òptima? Com influeix la inicialització del model en el resultat final?

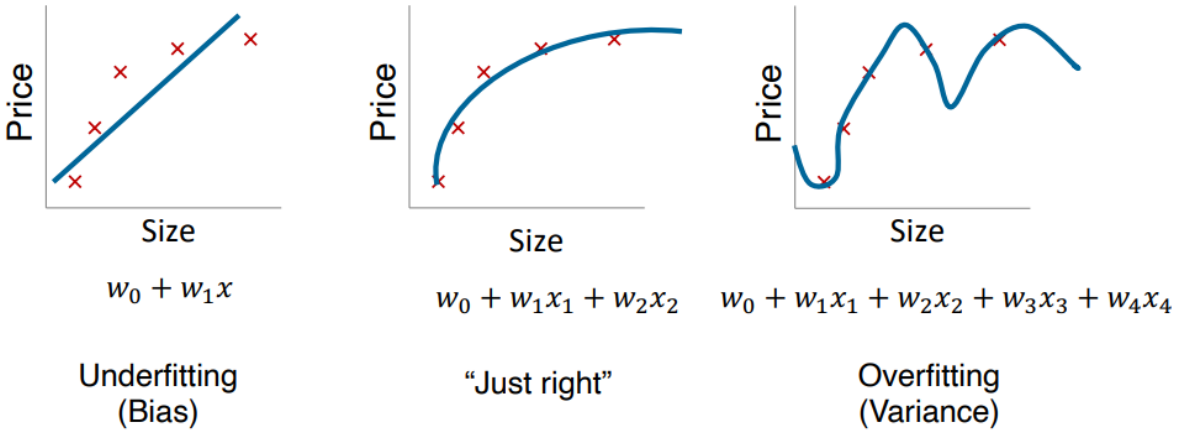
- **El coeficient d'aprenentatge α**: Aquest és un paràmetre que decideix en gran part el valor de la w. Hem d'anar amb compte a l'hora d'assignar aquest valor, ja que si assignem un molt gran pot passar això:



Com podem observar les w donen salts tants grans que podria donar-se el cas de mai trobar un mínim. La funció no convergiria i en alguns casos no divergiria, cosa que no volem. Per altra part, si el coeficient és molt petit, el descens tardaria molt a trobar una solució, els salts que dona la w serien molt petits com es veu en aquest exemple:



- **El regularitzador** λ:Aquest és el que penalitza models molt complexos, és a dir, penalitza graus molt alts de polinomis. Això és per evitar l'overfitting, donant tanta llibertat al polinomi que és massa igual a les dades entrades de test sent incapaç de predir dades noves. El contrari també pot passar (underfitting), per això trobar un paràmetre correcte es tan important com podem veure en aquest exemple:



- **El paràmetre Ɛ:** Aquest no apareix directament a la fórmula, però és el que decideix en gran part si el cost està convergint. És un valor que defineix la diferència mínima que ha de tenir el cost anterior i l'actual per tal de considerar els dos costos "propers". Un valor molt alt farà que la funció convergeixi molt ràpidament tot i no ser molt exacte.

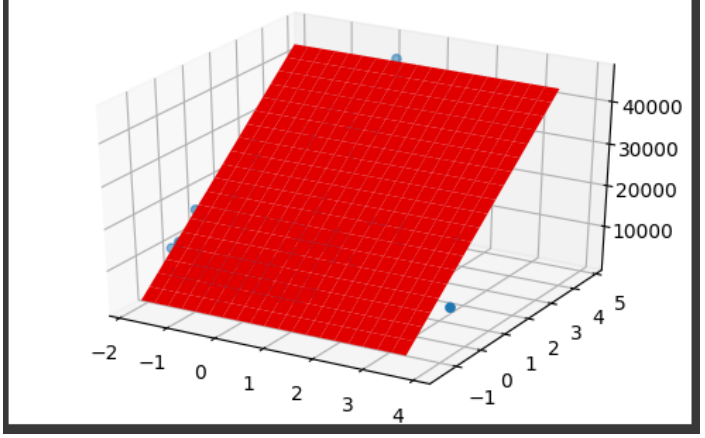
La inicialització pot influir en el resultat final, ja que, depenent d'aquesta, el descens acabarà anant cap a un mínim o cap a altres. És per això que s'han de provar molts valors d'inicialització per trobar un que arribi al cost més petit possible.

2. Quines funcions polinomials (de diferent grau, de diferents combinacions d'atributs, ...) heu escollit per ser apreses amb el vostre descens del gradient? quina ha donat el millor resultat (en error i rapidesa en convergència)?

En el nostre cas hem optat per escollir els valors que més relacionat estaven amb l'atribut objectiu, que és el preu. Ens hem fixat en un heatmap de l'apartat B.1 que representa la correlació entre tots els atributs. Per no generar overfitting vam pensar a fer un polinomi de grau 1 a 3:

- Primerament vam agafar l'atribut més relacionat: "Engine Size" cosa que ens va mostrar les següents dades representades en un espai 3D:

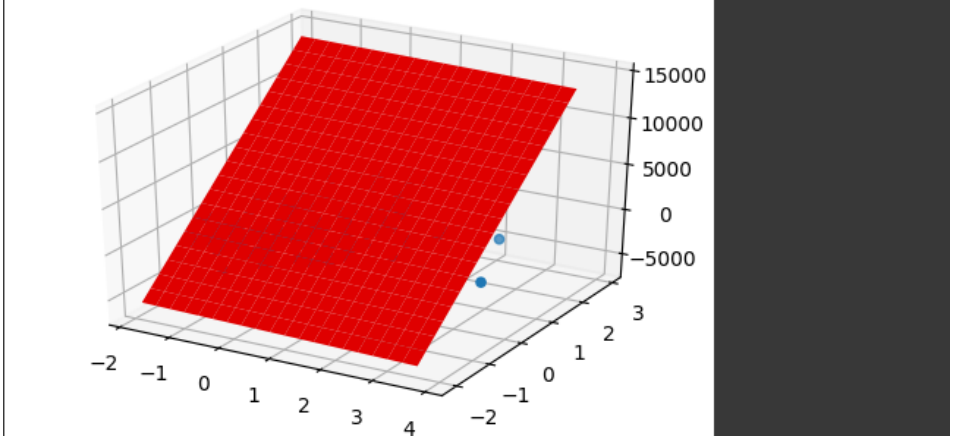
```
Init values: W0: 1.278345158631893 W1: 0.07538912066139358
Total iterations: 206841
Best cost is: 7_518_760.357890498
Best Ws were: W0: 13141.747512438817 W1: 6881.005044669275
Attributes selected for W1 and forth were: ['engine_size']
<mpl_toolkits.mplot3d.art3d.Path3DCollection at 0x7faca2b1dc10>
```



El cost és de 7 milions que ja millora el descens de la llibreria scikit, però encara volíem trobar millors resultats.

- La següent prova la vam fer amb un polinomi de grau 2 on les w1 i w2 van ser “curb_weight” i “Engine Size”

```
Init values: W0: 1.278345158631893 W1: -0.01485764173587552 W2: 0.07538912066139358
Total iterations: 298338
Best cost is: 7_305_803.058891604
Best Ws were: W0: 12244.542439290306 W1: 5444.2462765717555 W2: 901.6687812499653
Attributes selected for W1 and forth were: ['curb_weight' 'engine_size']
<mpl_toolkits.mplot3d.art3d.Path3DCollection at 0x7faca3074d50>
```



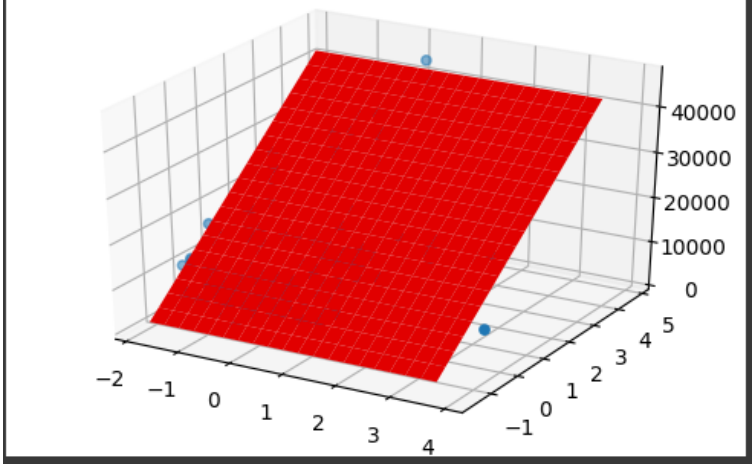
Amb això aconseguim un cost menor tot i augmentar un grau el polinomi, cosa que millora el nostre model

- Fent la prova amb més de dos graus fa que el cost augmenti més i, per tant, el millor trobat és l'anterior, ja que aquest amb grau 3 aconsegueix un cost de 7.799.158' 452909959. Aquest cost es troba amb els atributs w1, w2 i w3 de "horsepower", "curb_weight" i "engine_size" respectivament.

3. Utilitzeu el regularitzador en la fòrmula de funció de cost i descens del gradient i proveu polinomis de diferent grau. Com afecta el valor del regularitzador?

El valor del regularitzador, com està explicat a l'apartat A.1, és el que penalitza polinomis amb molts graus.
En el cas nostre aquest valor és de 0.01, ja que sembla que és aquell valor que no penalitza massa polinomis amb graus petits (graus 1-3) però tampoc deixa marge a passar-se amb els graus. És un valor que sembla estar en el punt correcte.
Jugant amb aquest, si posem un valor gran de 10 per exemple fa molt poques iteracions i no arriba a trobar costos petits com podem veure en aquests valors trobats:

```
Init values: W0: 1.278345158631893 W1: 0.07538912066139358
Total iterations: 32
Best cost is: 12_822_240.686239788
Best Ws were: W0: 12743.008226987606 W1: 6672.207555084623
Attributes selected for W1 and forth were: ['engine_size']
<mpl_toolkits.mplot3d.art3d.Path3DCollection at 0x7faca2282c10>
```



En canvi, posant un valor molt petit les iteracions són molt altes i estarem acceptant polinomis amb graus molt petits que serien massa simples.

4. Quina diferència (quantitativa i qualitativa) hi ha entre el vostre regressor i el de la llibreria ?

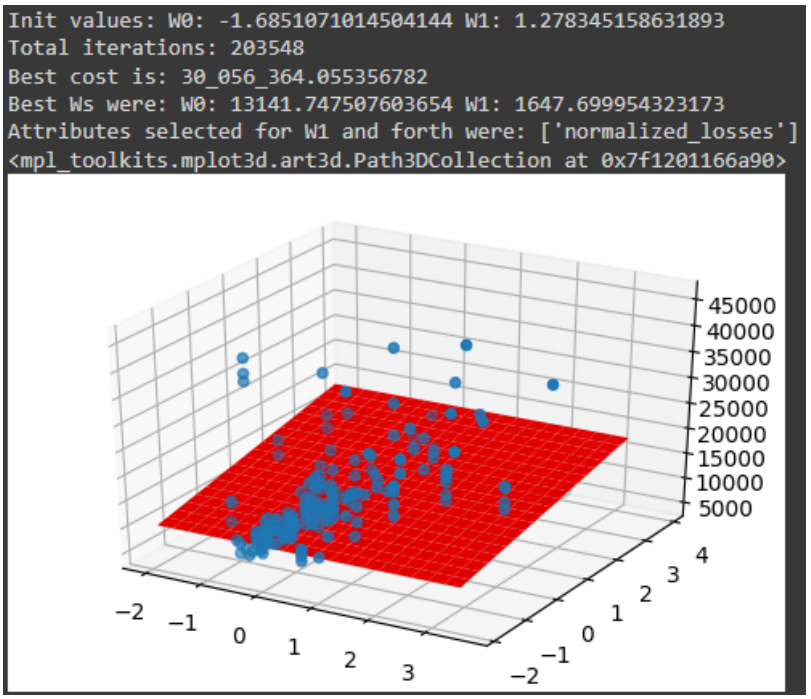
La diferencia principal és que amb el regressor de la llibreria hem de fer PCA per tal de reduir les dimensions i poder representar els valors amb una recta.
Això no estaria malament si no afectés el resultat, però al reduir les dimensions perdem exactitud. A més no ho podem personalitzar massa. En canvi, amb el regressor implementat podem fer proves i canviar els diferents paràmetres com està explicat a l'apartat A.1 per trobar el cost mínim.
És per això que el resultat donat té un cost menor que el que la recta generada per la llibreria, cosa que indica que el nostre model prediu millor futures dades.

5. Té sentit el model (polinomial) trobat quan es visualitza sobre les dades?

Quan el model és de grau 1 sí que té sentit la visualització de les dades, ja que aquesta representa una recta sobre tots els punts de la y. Quan el model és de grau superior no estaríem visualitzant correctament el pla, ja que ens faria falta més dimensions per representar. Tot i això podem fer-nos una idea de quan bo és el model amb aquestes dimensions i l'ús del cost per trobar la qualitat de la funció.

6. Ajuda la visualització a identificar aquelles mostres per a les que el regressor obté els pitjors resultats de predicció?

Sí que ens ajuda, ja que podem veure d'una manera clara les funcions que no tenen molt de sentit. Es pot identificar fàcilment que els valors que predirà respecte a els reals són molt dispars. Com a exemple, he posat un valor molt poc correlacionat a l'hora de fer el descens de gradient i ha sortit el següent:



Com podem observar no és un bon model i no solament pel cost excessiu de 30M sinó perquè es veu clarament que el pla no està gens proper als punts correctes.