Simulació Física [LR]

Integrants del grup

- **Alex Pons** (alex.pons)
- **Joel López** (joel.lopez)

Enunciat

Aquesta pràctica té com a objectiu fer una mirada més profunda al ***Machine Learning***, i executar un estudi sobre una base de dades en format *.csv*.

Consisteix en agafar valors de testegi i fer un anàlisis de predicció mitjançant una recta de regressió lineal, estudiar els valors de cost i θ's mentre el sistema aprèn, així com trobar els *outliers* d'aquest *dataset*, eliminar-los i, per últim, executar-ho de nou, comparant ambdós resultats.

Estructura de la entrega

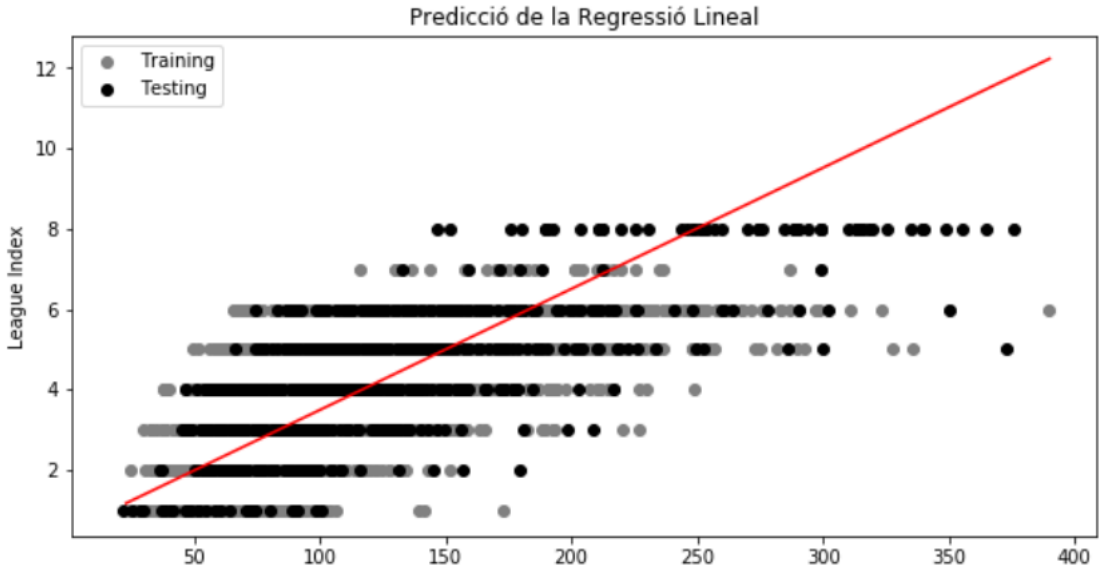
L'arxiu *.ipynb* de python que executarem està organitzat en els punts i ordre següent:

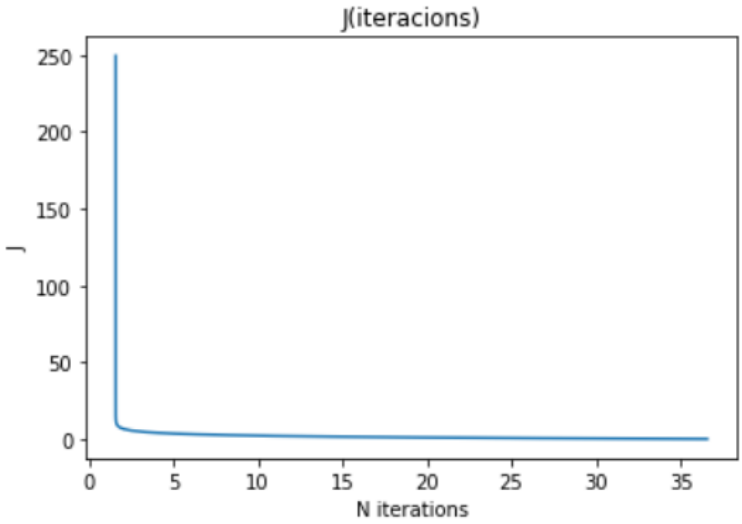
1. **Imports**
2. **Dataset info**
3. **Variables**
    a. Training and testing data
    b. Related to computin that data (n, m, altres arrays...)
4. **Functions' definition**
    a. Slope function (funció de la recta de regressió lineal)
    b. Gradient step (funció que calcula la recta gradient)
    c. Plot line (funció que mostra la gràfica de cost)
    d. Plot array (funció que crea una gràfica genérica)
    e. Current cost (funció que calcula el cost)
    f. Detect outlier (funció que troba els outliers -amb el Z-score anomenat anteriorment)
5. **Model training**
6. **Graphs with starting training data**
7. **Getting rid of outliers**
    a. Busquem els outliers i creem una matriu que contingui els índexs d'aquests
    b. Els esborrem i creem un nou array de valors de apm (x) i lliga (y) sense aquests: **x_train_nooutliers** i **y_train_nooutliers**
8. **Training our model with the new data**
9. **Show graphs of our new data**
10. **Result's comparison (a continuació)**

Exercici 6. Comparació de dades amb i sense outliers

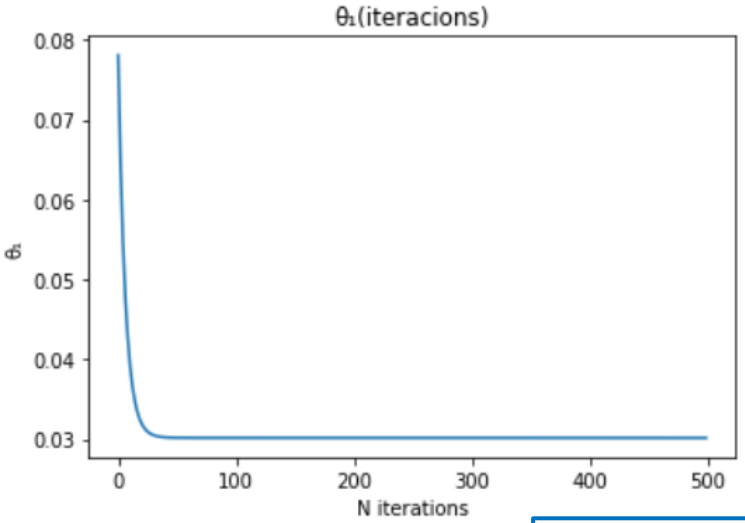
Primerament mostrarem les gràfiques resultants de l'aprenentatge del nostre sistema amb les dades d'entrenament que vàrem escollir (el 80% de totes les dades),

de dalt a baix, el gràfic genèric amb la recta de regressió, la funció de cost i l'evolució de θ_1. A continuació, les mateixes un cop hem trobat els outliers i els hem extret:





Si fem zoom amb la funció **xlim**, podem apreciar que, efectivament, la forma de la recta coincideix amb l'expectativa d'avançar mentre aprèn fins arribar a una cota de cost determinada.



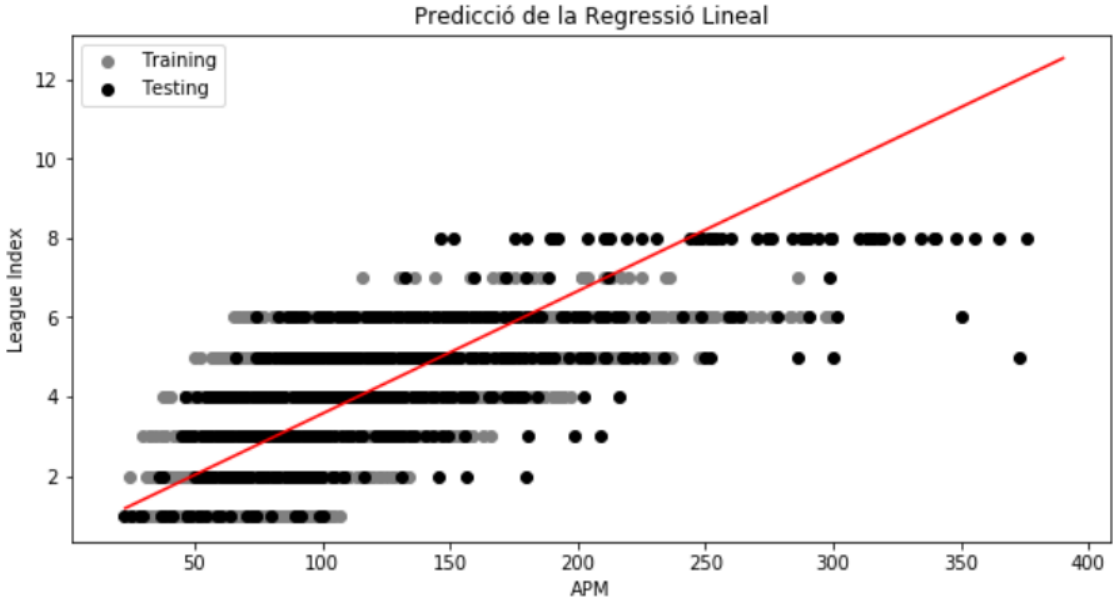
Partint d'un valor aleatori, com manava l'enunciat, podem veure que, a mesura que el sistema aprèn, la velocitat a la que el nostre valor θ_1 canvia es va reduint fins arribar a una cota al valor final pres fins a les N iteracions que hem definit preferentment (al nostre cas, 500 iteracions a un ritme d'aprenentatge de 0.0001, ja que θ_1 acaba prenent un valor molt petit:

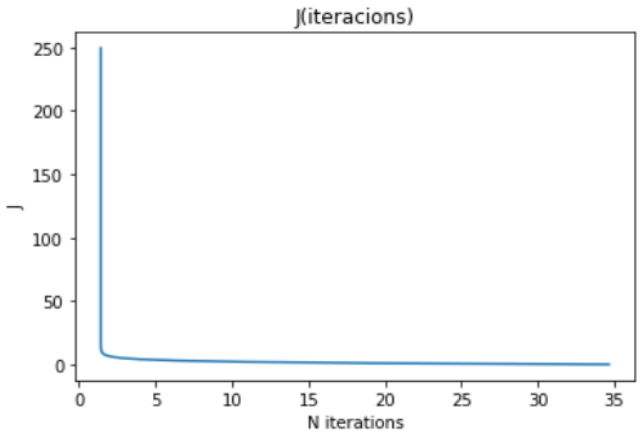
```
Value of θ₀ (n): 0.47780125357141146
Value of θ₁ (m): 0.030146623941572224
```

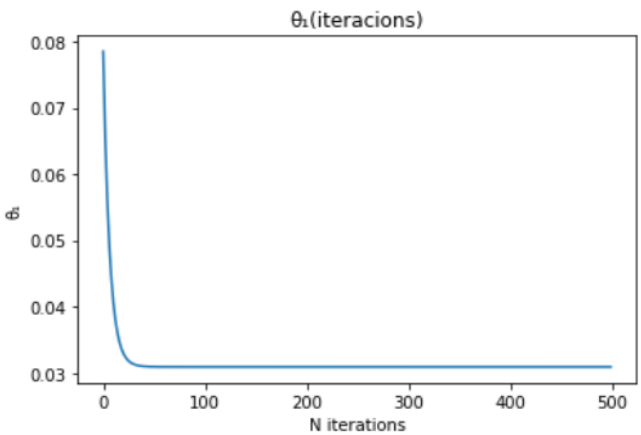
Després d'aquest anàlisi inicial, detectem els outliers i els extraiem del dataset abans de tornar-ho a estudiar:

```
-> League 1: 140 elements.
        3 outlier(s) detected!
-> League 2: 279 elements.
        2 outlier(s) detected!
-> League 3: 441 elements.
        7 outlier(s) detected!
-> League 4: 661 elements.
        10 outlier(s) detected!
-> League 5: 653 elements.
        9 outlier(s) detected!
-> League 6: 514 elements.
        3 outlier(s) detected!
-> League 7: 28 elements.
        0 outlier(s) detected!
```

```
Liga 1 -> Eliminado valor 141.6282 (indice = 127)
Liga 1 -> Eliminado valor 172.953 (indice = 124)
Liga 1 -> Eliminado valor 139.6362 (indice = 102)
Liga 2 -> Eliminado valor 142.767 (indice = 272)
Liga 2 -> Eliminado valor 151.64700000000002 (indice = 169)
Liga 3 -> Eliminado valor 193.3422 (indice = 435)
Liga 3 -> Eliminado valor 182.3736 (indice = 377)
Liga 3 -> Eliminado valor 192.3036 (indice = 348)
Liga 3 -> Eliminado valor 188.1348 (indice = 286)
Liga 3 -> Eliminado valor 226.6554 (indice = 268)
Liga 3 -> Eliminado valor 189.465 (indice = 177)
Liga 3 -> Eliminado valor 220.0692 (indice = 85)
Liga 4 -> Eliminado valor 211.7124 (indice = 660)
Liga 4 -> Eliminado valor 249.021 (indice = 655)
Liga 4 -> Eliminado valor 227.2272 (indice = 638)
Liga 4 -> Eliminado valor 215.01 (indice = 605)
Liga 4 -> Eliminado valor 214.2684 (indice = 526)
Liga 4 -> Eliminado valor 216.4152 (indice = 498)
Liga 4 -> Eliminado valor 206.838 (indice = 466)
Liga 4 -> Eliminado valor 229.9122 (indice = 433)
Liga 4 -> Eliminado valor 209.8026 (indice = 221)
Liga 4 -> Eliminado valor 216.6936 (indice = 6)
Liga 5 -> Eliminado valor 289.4268 (indice = 555)
Liga 5 -> Eliminado valor 274.9086 (indice = 494)
Liga 5 -> Eliminado valor 281.4246 (indice = 416)
Liga 5 -> Eliminado valor 272.8404 (indice = 408)
Liga 5 -> Eliminado valor 292.5408 (indice = 333)
Liga 5 -> Eliminado valor 256.8528 (indice = 327)
Liga 5 -> Eliminado valor 254.5392 (indice = 279)
Liga 5 -> Eliminado valor 327.7218 (indice = 57)
Liga 5 -> Eliminado valor 335.499 (indice = 29)
Liga 6 -> Eliminado valor 323.2506 (indice = 308)
Liga 6 -> Eliminado valor 311.0094 (indice = 170)
Liga 6 -> Eliminado valor 389.8314 (indice = 127)
```







A primera vista no sembla que hi hagi cap canvi apreciable, però si comparem els resultats exactes que ens ofereix el sistema un cop ha calculat de nou la forma de la recta de regressió:

```
Value of θ₀ (n) [NO OUTLIERS]: 0.4776126261152944
Value of θ₁ (m) [NO OUTLIERS]: 0.030920044363469457
```

Veiem que θ_1 ha cambiat el seu valor en una quantitat de **0.0008 unitats**. Ni tans sols arriba a un error mil·limètric! Per tant, la recta de regressió pràcticament no ha patit canvis.

Explicació

Els motius més lògics pels quals ha passat això són dos:

- **La quantitat de dades preses.** Un cop agafat el 80% de les dades totals per a l'entrenament del sistema, ens hem quedat amb una quantitat d'unes 2700 dades aproximadament. Més tard, en total hem extret uns 30 outliers. Això vol dir que pràcticament ens hem quedat igual amb els darrers valors, i, per tant, els valors de *theta* no han patit pràcticament cap canvi per la mínima informació extreta en comparació amb la restant, tot i ser outliers.
- **Els valors de Y.** Aquesta potser és menys directa, però, si ho pensem bé, el data set que hem fet servir tenia molt poca variació de valors a l'eix d'ordenades (només de 1 a 8), mentre que tenim un munt de variació a l'eix d'abscisses (centenars). Això implica que la pendent de la recta és molt marcat en quant a l'avanç de les X però més ràpid a les Y (per això la *theta* que acompanya a la x a la fórmula és bastant més petit que la *theta 0*). Al haver molta més distribució d'informació a les X, els outliers tenen molta menys presència quan pràcticament tots els valors de cada lliga son els mateixos si, A MÉS, hi ha poques lligues, com hem dit. En el cas d'haver més lligues, l'avenç de la recta seria més pronunciat, hi hauria menys informació (menys X) per a cada valor de les Y, el que implica que els outliers prendrien més força i, per tant, la recta seria una mica més diferent.

Val la pena extreure els outliers, llavors?

Segons l'exactitud de les dades que es vulguin obtenir. En cas de fer servir aquestes mateixes dades, no era necessari, ja que el temps implicat en el segon estudi del sistema i els canvis resultants són massa distants. *"No val la pena si en un valor de X que varia en centenars i un de Y que varia en unitats, l'error que es comet és inferior al 0.001".*

Eines addicionals

A més dels imports i les funcions recomanades a l'enunciat, també hem fet ús d'altres que hem trobat a mesura que investigàvem:

- Tot i que una opció seria haver agafat la part corresponent de l'anterior pràctica (ML0), per trobar els **outliers** hem fet servir un altre mètode que vàrem descobrir investigant, anomenat **Z-score**. En resum, aquest consisteix en una sèries d'operacions de distribució de dades (semblant als quartiles, però sense comparar els seus valors), trobant un valor de semblança entre

ells, i després fent servir un **threshold** (habitualment **3** o **-3**) per considerar els possibles outliers de la llista oferida. Per això hem agregat una altra llibreria anomenada **"stats"**, de **scipy**.
Després de fer servir aquest mètode hem comprovat que els resultats son els mateixos que els que podem observar al fer un **boxplot** de la llibreria **matplotlib**.

- ➔ Tot i que al final no els hem deixat a l'entrega final, a l'hora de testejar, per observar millor els resultats hem fet servir els **xlim** i **ylim** per tallar els eixos de coordenades al generar els gràfics quan provàvem d'estudiar les gràfiques i comparar els resultats (sobretot perquè els valors de θ són molt propers a zero).
- ➔ Si fem servir un array definit inicialment com a var = [], el sistema triga aproximadament un segon en fer cada iteració. Si, en canvi, fem servir la funció **tolist()** del **csv_reader**, la llista resultant és infinitament més ràpida.
- ➔ Hem utilitzat la funció **random()** de la llibreria **numpy** per agafar els valors inicials de **m** i **n**.
- ➔ **p**k** (número **p elevat a k**) i **abs()** pel tractament de la informació.
- ➔ Funció **scatter()**, per generar la gràfica de predicció.
- ➔ A l'hora d'extreure els outliers, vam haver de recórrer l'array a la inversa, ka que si esborràvem l'índex (per exemple) 5 de la llista de dades d'entrenament, tot l'array restant s'avançava en una posició. En comptes, si esborràvem primer (suposem també) el 10 i després el 5, el desplaçament de l'array no esdevé un problema (evidentment, perquè la llista d'índexs que representaven els outliers estava ordenada de menor a major). Per això hem utilitzat la funció **reverse()**, implícita de python.
- ➔ També referent a l'analisi de les dades i l'estudi del arrays, hem fet ús de la funció **np.where()**, per trobar els outliers mitjançant l'expressió del **Z-score**, com hem dit abans; i **pop()**, del tipus llista, per extreure'ls.
- ➔ Funció **describe()** i **numpy.percentile()** per comprovar les dades (mediana, first/third quartiles, etc.) [https://stackoverflow.com/questions/45926230/how-to-calculate-1st-and-3rd-quartiles]

Conclusions

Tot i que semblava un treball difícil, hem trobat un munt d'informació disponible en internet sobre aquests temes (i, sorprenentment, quasi tot en el mateix llenguatge de programació, python!), i hem après vàries maneres d'analitzar la informació (com, per exemple, el Z-score). La pràctica no era tan difícil com en un principi semblava, però hem tingut una sorprenent massa de problemes, especialment amb l'anàlisi dels outliers i les divisions per lligues. Alguns dels quals potser hauríem solucionat fent servir matrius per el dataset, dividit en lligues i després en jugadors, com hem acabat fent servir més endavant per estudiar els outliers, tot i que no se'ns va ocórrer fins més endavant.

Ha estat interessant veure com hem creat un sistema que aprèn per primera vegada. Seria curiós aplicar-ho als videojocs, per exemple!