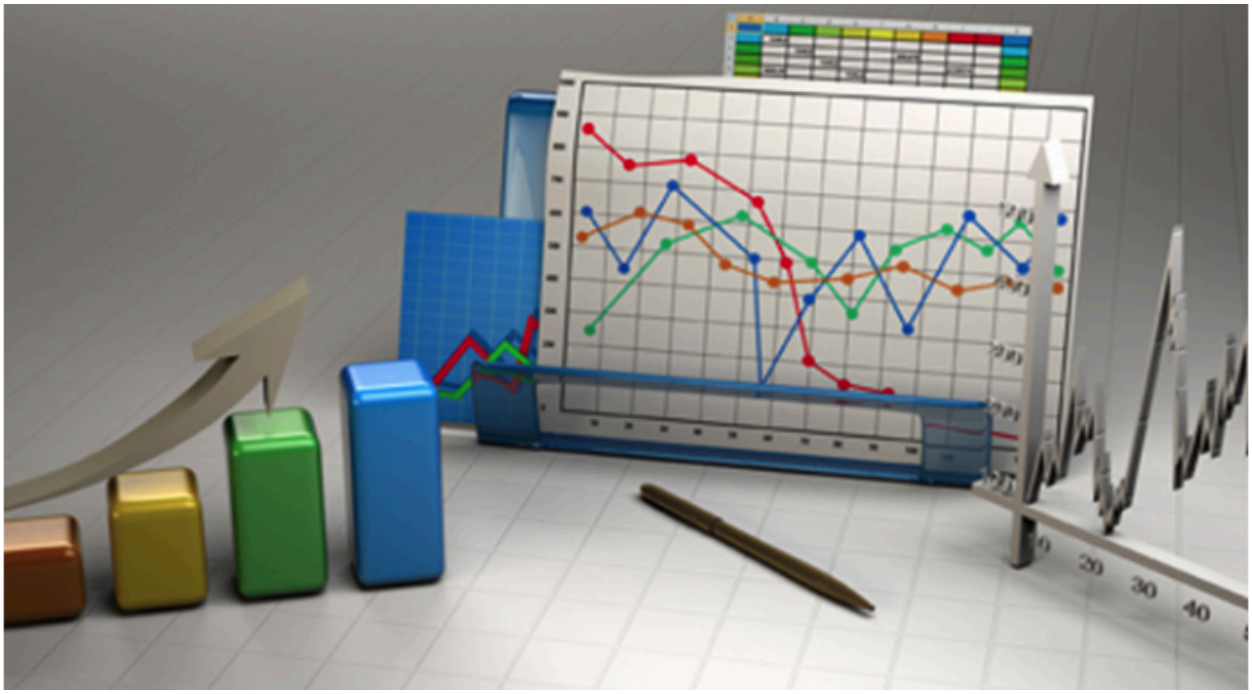


ENTREGA REGRESIÓN LINEAL 2024 - 2C

Optimización de Presupuesto en el Mercado de Jugadores del FIFA 21 para la Empresa PANINOS

—

Integrantes:

Heredia Pamela 18179/1

García Camila 12990/6

Parte 1: Predicción del valor de mercado

a. Recta de regresión para predecir el valor de mercado de un jugador a partir de la característica más relevante (a la que se destinará mayor proporción del presupuesto), respaldada por:
   i. Prueba de significancia de regresión, coeficiente de determinación (R^2) y correlación lineal (r).
   ii. Inferencias sobre los parámetros de la recta, estimando las fluctuaciones con una confianza del 95%.
   iii. La proporción de veces que el valor de mercado supera la incertidumbre de predicción comparada con la respuesta media del valor de mercado para una característica fija, ambas con la misma confianza y ancho mínimo.

Se nos pide encontrar la característica que mejor nos ayude a predecir el valor de mercado de un jugador (la cual será la variable dependiente y). Entonces, lo que debemos hacer es obtener la recta de regresión lineal para cada una de las variables que se nos provee en el archivo players_21.csv

Hay que tener en cuenta que no todas las variables que representan las columnas nos van a servir para construir el modelo, ya que hay valores que son información no numérica o que no es relevante para lo que buscamos, como por ejemplo: Nacionalidad, Equipo, nombre (si consideramos nombre= Lionel messi se rompen todas las métricas 😂), etc.

Además, hay columnas que pueden ser de interés, pero que para algunos jugadores no hay valores definidos, por lo que podemos elegir no usar dichas columnas, o rellenar con ceros esos valores(o enmascararlos). En nuestro caso elegimos no usar dichas columnas para el modelo, ya que el hecho de no tener datos en esas métricas no representa que sea algo bueno o malo, lo cual puede sesgar nuestras predicciones.

Una vez depurado nuestro source, y con los datos seleccionados, podremos obtener no sólo la recta de regresión (que servirá para hacer predicciones), sino que podemos calcular los coeficientes de determinación y correlación lineal, que son los que nos darán una pauta de qué tan buenas serán esas predicciones, y qué tan fuerte es su relación lineal con los valores obtenidos (a mayor linealidad, mejores resultados).

Al ejecutar el código en python, obtenemos los estimadores para cada una de las características provistas:

```
Característica: age
  β₀ (intercepto): -103574.12
  β₁ (pendiente): 92301.74
  SCE (Suma de Cuadrados del Error): 489626860774378624.00
  STC (Suma Total de Cuadrados): 493187887278613056.00
------------------------------
Característica: height_cm
  β₀ (intercepto): 1666228.95
  β₁ (pendiente): 3082.85
  SCE (Suma de Cuadrados del Error): 493179499538721664.00
  STC (Suma Total de Cuadrados): 493187887278613056.00
------------------------------
Característica: weight_kg
  β₀ (intercepto): -200212.25
  β₁ (pendiente): 32326.39
  SCE (Suma de Cuadrados del Error): 492202014726506816.00
  STC (Suma Total de Cuadrados): 493187887278613056.00
------------------------------
Característica: overall
  β₀ (intercepto): -27930247.75
  β₁ (pendiente): 459136.37
  SCE (Suma de Cuadrados del Error): 297388715814738944.00
  STC (Suma Total de Cuadrados): 493187887278613056.00
------------------------------
Característica: potential
  β₀ (intercepto): -31727509.92
  β₁ (pendiente): 477618.31
  SCE (Suma de Cuadrados del Error): 331866860283656960.00
  STC (Suma Total de Cuadrados): 493187887278613056.00
------------------------------
Característica: wage_eur
  β₀ (intercepto): 332598.82
  β₁ (pendiente): 218.1
  SCE (Suma de Cuadrados del Error): 145090065841134720.00
  STC (Suma Total de Cuadrados): 493187887278613056.00
------------------------------
Característica: international_reputation
  β₀ (intercepto): -6461881.06
  β₁ (pendiente): 7955943.62
  SCE (Suma de Cuadrados del Error): 336199736998743680.00
  STC (Suma Total de Cuadrados): 493187887278613056.00
------------------------------
Característica: weak_foot
```

```
  β₀ (intercepto): -1432664.99
  β₁ (pendiente): 1245479.47
  SCE (Suma de Cuadrados del Error): 480109776434861632.00
  STC (Suma Total de Cuadrados): 493187887278613056.00
------------------------------
Característica: skill_moves
  β₀ (intercepto): -2395709.43
  β₁ (pendiente): 1955348.65
  SCE (Suma de Cuadrados del Error): 450639195603128192.00
  STC (Suma Total de Cuadrados): 493187887278613056.00
------------------------------
```

I. Recta de Regresión

Una vez obtenidos los estimadores, podemos calcular las rectas de regresión, junto con los coeficientes de determinación y de correlación; para luego compararlos y decidir cuál es la mejor métrica para predecir el valor de un jugador en el mercado.

Entonces, la ecuación de la línea de mínimos cuadrados de cada variable resulta en:

$$\widehat{y_{feature}} = \widehat{\beta_{0_{selected_{feature}}}} + \widehat{\beta_{1_{selected_{feature}}}} \cdot x$$

1. $\widehat{y_{age}} = \widehat{\beta_{0_{age}}} + \widehat{\beta_{1_{age}}} \cdot x$
   - $\widehat{y_{age}} = -103574.12 + 92301.74\,x$
2. $\widehat{y_{height\_cm}} = \widehat{\beta_{0_{height\_cm}}} + \widehat{\beta_{1_{height\_cm}}} \cdot x$
   - $\widehat{y_{height\_cm}} = 1666228.95 + 3082.85 \cdot x$
3. $\widehat{y_{weight\_kg}} = \widehat{\beta_{0_{weight\_kg}}} + \widehat{\beta_{1_{weight\_kg}}} \cdot x$
   - $\widehat{y_{weight\_kg}} = -200212.25 + 32326.39 \cdot x$
4. $\widehat{y_{overall}} = \widehat{\beta_{0_{overall}}} + \widehat{\beta_{1_{overall}}} \cdot x$
   - $\widehat{y_{overall}} = -27930247.75 + 459136.37\,.x$
5. $\widehat{y_{potential}} = \widehat{\beta_{0_{potential}}} + \widehat{\beta_{1_{potential}}} \cdot x$

- $\widehat{y_{potential}} = -31727509.92 + 477618.31 \cdot x$

6. $\widehat{y_{wage\_eur}} = \widehat{\beta_0}_{wage\_eur} + \widehat{\beta_1}_{wage\_eur} \cdot x$
   - $\widehat{y_{wage\_eur}} = 332598.82 + 218.1 \cdot x$
7. $\widehat{y_{international\_reputation}} = \widehat{\beta_0}_{international\_reputation} + \widehat{\beta_1}_{international\_reputation} \cdot x$
   - $\widehat{y_{international\_reputation}} = -6461881.06 + 7955943.62 \cdot x$
8. $\widehat{y_{weak\_foot}} = \widehat{\beta_0}_{weak\_foot} + \widehat{\beta_1}_{weak\_foot} \cdot x$
   - $\widehat{y_{weak\_foot}} = -1432664.99 + 1245479.47 \cdot x$
9. $\widehat{y_{skill\_moves}} = \widehat{\beta_0}_{skill\_moves} + \widehat{\beta_1}_{skill\_moves} \cdot x$
   - $\widehat{y_{skill\_moves}} = -2395709.43 + 1955348.65 \cdot x$

Coeficiente de Determinación

Con la información calculada anteriormente, obtenemos los coeficientes de determinación, que explican el porcentaje de información explicado por la recta de regresión para cada una de las características. El coeficiente de determinación está definido de la siguiente forma:

$$R^2_{feature} = 1 - \frac{SCE_{selected_{feature}}}{STC_{selected_{feature}}}$$

Coeficiente de Correlación

Y considerando que para verificar si la relación lineal es la más correcta, se debe calcular el coeficiente de correlación lineal que para cada característica está dado por:

$$r_{feature} = \sqrt{R^2_{feature}}$$

Realizando los cálculos obtenemos que los coeficientes son:

```
Característica: age
```

```
  Coeficiente de Correlación Lineal: 0.08
  Coeficiente de Determinación: 0.01
-----------------------------
Característica: height_cm
  Coeficiente de Correlación Lineal: 0.00
  Coeficiente de Determinación: 0.00
-----------------------------
Característica: weight_kg
  Coeficiente de Correlación Lineal: 0.04
  Coeficiente de Determinación: 0.00
-----------------------------
Característica: overall
  Coeficiente de Correlación Lineal: 0.63
  Coeficiente de Determinación: 0.40
-----------------------------
Característica: potential
  Coeficiente de Correlación Lineal: 0.57
  Coeficiente de Determinación: 0.33
-----------------------------
Característica: wage_eur
  Coeficiente de Correlación Lineal: 0.84
  Coeficiente de Determinación: 0.71
-----------------------------
Característica: international_reputation
  Coeficiente de Correlación Lineal: 0.56
  Coeficiente de Determinación: 0.32
-----------------------------
Característica: weak_foot
  Coeficiente de Correlación Lineal: 0.16
  Coeficiente de Determinación: 0.03
-----------------------------
Característica: skill_moves
  Coeficiente de Correlación Lineal: 0.29
  Coeficiente de Determinación: 0.09
```

Sabemos que:

- El coeficiente de correlación lineal toma valores entre -1 y 1 en donde:
    - r = –1 es una correlación perfecta negativa.
    - r = 0 implica que la correlación es nula.
    - r = 1 es una correlación positiva perfecta

- El coeficiente de determinación, que mide la calidad del modelo, toma valores entre 0 y 1 donde:
    - Valores cercanos a 0 indican que el modelo no es bueno.
    - Valores cercanos a 1 indican que el modelo es bueno para predecir el comportamiento de la variable dependiente.

Entonces la conclusión a partir de los datos obtenidos, es que la recta estimada que mejor se ajusta a los datos y que mejor servirá para predecir el valor de mercado de un jugador será la siguiente:

- Característica: wage_eur
- Recta de regresión: $\widehat{y_{wage\_eur}} = 332598.82 + 218.1 \cdot x$
- Coeficiente de Correlación Lineal: 0.84 (cercano a 1)
- Coeficiente de Determinación: 0.71 (cercano a 1)

Esta característica y su correspondiente recta de regresión es la que mejor se ajusta a la predicción del valor de mercado de un jugador, porque sus coeficientes tanto de determinación como de correlación lineal son más cercanos a 1 en comparación con el resto de valores obtenidos de otras características.

Prueba de significancia de regresión:

Una vez obtenidas las rectas, podemos hacer las pruebas de significancia de las variables, empezando por la característica de wage_eur que es la que dió mejores valores en los coeficientes de determinación y correlación lineal calculados previamente.

Como lo que nos interesa es verificar la linealidad entre $x_{feature}$ e y, analizaremos el valor de β_1 para cada recta de regresión obtenida, de manera de comprobar si existe o no una relación entre ambas variables. Aceptar $H_0: \beta_1 = 0$, es equivalente a concluir que no hay ninguna relación lineal entre x e y. Caso contrario concluir que $x_{feature}$ tiene importancia al explicar la variabilidad en y.

Las hipótesis definidas son:

- $H_0: \beta_1 = 0$
- $H_1: \beta_1 \neq 0$

El estadístico de prueba es el siguiente, el cual bajo la hipótesis nula H_0 tiene distribución Student con $n - 2$ grados de libertad.

Como la hipótesis alternativa H_1 es una desigualdad, la regla es:

- Rechazar H_0 si $|T| > t_{\frac{\alpha}{2}, n-2}$
- No rechazar H_0 si $|T| \leq t_{\frac{\alpha}{2}, n-2}$

Al calcular T, obtenemos:

$$T = \frac{\widehat{\beta_1} - \theta}{\sqrt{\frac{\widehat{\sigma}^2}{S_{xx_{feature}}}}}$$

Donde:

- $S_{xx_{feature}}$ = 7189132432510.45
- Número de observaciones (n): 17949
- $\widehat{\sigma}^2$ = 7958226527145.8662

Reemplazamos por los valores obtenidos:

$$T = \frac{218.35 - (0)}{\sqrt{\frac{7958226527145.8662}{7189132432510.45}}} = \frac{218.35}{\sqrt{1.106}} = \frac{218.35}{1.052} = 207.55$$

Al no contar con un nivel de significancia especificado, tenemos que calcular el P-valor para una hipótesis de dos colas (por tratarse de una desigualdad) con $T = 207.55$ y $n - 2 = 17949 - 2 = 17947$ grados de libertad; al calcularlo obtuvimos un resultado que p-valor es < 0.00001.

También utilizamos código en python para confirmar los cálculos realizados.

Como un número menor a 0.00001, también es menor a 0.5, se rechaza H_0 y se acepta H_1.

Por lo tanto, significa que $x_{wage_{eur}}$ tiene importancia al explicar la variabilidad de y.

Conclusión

En conclusión, la elección de la característica wage_eur y su recta de regresión para estimar y predecir el valor de mercado de un jugador está respaldada por los coeficientes de correlación y determinación, y por la prueba de significancia de regresión.

II. Inferencias sobre los parámetros de la recta, estimando las fluctuaciones con una confianza del 95%.

Para este punto se pide calcular intervalos con una confianza del 95%.

Sabemos que el intervalo de confianza para $\beta_0$está dado por:

$$\left[\widehat{\beta_0} \mp t_{\frac{\alpha}{2}, n-2} \cdot \sqrt{\widehat{\sigma^2} \cdot \left(\frac{1}{n} + \frac{\overline{x_a}^2}{S_{x_a x_a}}\right)}\right]$$

Y el intervalo de confianza de β_1 es:

$$\left[\widehat{\beta_1} \mp t_{\frac{\alpha}{2}, n-2} \cdot \sqrt{\frac{\widehat{\sigma^2}}{S_{x_a x_a}}}\right]$$

Para obtenerlos necesitamos los siguientes datos (la mayoría ya los calculamos previamente):

- $\alpha = 0.05$
- n = 17949
- $grados_{libertad} = n - 2$ = 17947
- $t_{\frac{\alpha}{2}, n-2}$ = 1.9601
- S_{xx} = 7189132432510.45
- $\widehat{\sigma^2}$ = 7958226527145.8662
- $\widehat{\beta_0}$ = 332598.82
- $\widehat{\beta_1}$ = 218.1

Una vez que tenemos todos esos datos, podemos usar un script de python para calcular los intervalos de confianza para β_0 y β_1:

Intervalo de confianza al 95% para β1: (216.29, 220.41)

Intervalo de confianza al 95% para β0: (287467.58, 377730.06)

III. La proporción de veces que el valor de mercado supera la incertidumbre de predicción comparada con la respuesta media del valor de mercado para una característica fija, ambas con la misma confianza y ancho mínimo.

Lo que tenemos que hacer es calcular una proporción en función del error entre el intervalo de predicción, y el intervalo de confianza para la media de nuestra función.

Generalmente sucede que el intervalo de predicción tiene mayor error que el de respuesta media porque son dos variables aleatorias, y naturalmente el intervalo sobre una predicción tiene mayor error que el intervalo para un valor medio de datos ya medidos.

Pero podemos ver qué proporción existe para este caso con las siguientes fórmulas, aprovechando que $t_{\frac{\alpha}{2}, n-2}$ tienen el mismo valor en el numerador y denominador, porque el nivel de confianza es el mismo. Sabemos que dicha proporción se define de la siguiente forma:

$$\frac{\varepsilon(IP)}{\varepsilon(ICM)} = \frac{t_{\frac{\alpha}{2}, n-2} \cdot \sqrt{\widehat{\sigma^2} \cdot \left(1 + \frac{1}{n} + \frac{\left(x^* - \bar{x}\right)^2}{S_{xx}}\right)}}{t_{\frac{\alpha}{2}, n-2} \cdot \sqrt{\widehat{\sigma^2} \cdot \left(\frac{1}{n} + \frac{\left(x^* - \bar{x}\right)^2}{S_{xx}}\right)}}$$

Podemos operar y obtener así la expresión:

1. $$\frac{1 \cdot \sqrt{\widehat{\sigma^2} \cdot \left(1 + \frac{1}{n} + \frac{\left(x^* - \bar{x}\right)^2}{S_{xx}}\right)}}{1 \cdot \sqrt{\widehat{\sigma^2} \cdot \left(\frac{1}{n} + \frac{\left(x^* - \bar{x}\right)^2}{S_{xx}}\right)}}$$

2. $$\frac{\sqrt{\widehat{\sigma^2} \cdot \left(1 + \frac{1}{n} + \frac{\left(x^* - \bar{x}\right)^2}{S_{xx}}\right)}}{\sqrt{\widehat{\sigma^2} \cdot \left(\frac{1}{n} + \frac{\left(x^* - \bar{x}\right)^2}{S_{xx}}\right)}}$$

3. $$\sqrt{\frac{\widehat{\sigma^2} \cdot \left(1 + \frac{1}{n} + \frac{\left(x^* - \bar{x}\right)^2}{S_{xx}}\right)}{\widehat{\sigma^2} \cdot \left(\frac{1}{n} + \frac{\left(x^* - \bar{x}\right)^2}{S_{xx}}\right)}}$$

4. $$\sqrt{\frac{1 \cdot \left(1 + \frac{1}{n} + \frac{\left(x^* - \bar{x}\right)^2}{S_{xx}}\right)}{1 \cdot \left(\frac{1}{n} + \frac{\left(x^* - \bar{x}\right)^2}{S_{xx}}\right)}}$$

5. $$\sqrt{\frac{1 + \frac{1}{n} + \frac{\left(x^* - \bar{x}\right)^2}{S_{xx}}}{\frac{1}{n} + \frac{\left(x^* - \bar{x}\right)^2}{S_{xx}}}}$$

Definimos la proporción del error entre el intervalo de predicción y el intervalo de confianza de la media, y dado que el ancho es mínimo, tendremos que $\frac{\left(x^* - \bar{x}\right)^2}{S_{xx}} = 0$. Con lo cual podemos seguir operando en la ecuación y llegar a:

1. $$\sqrt{\frac{1 + \frac{1}{n} + \frac{(0)^2}{S_{xx}}}{\frac{1}{n} + \frac{(0)^2}{S_{xx}}}}$$

2. $$\sqrt{\frac{1 + \frac{1}{n} + \frac{0}{S_{xx}}}{\frac{1}{n} + \frac{0}{S_{xx}}}}$$

3. $$\sqrt{\frac{1 + \frac{1}{n} + 0}{\frac{1}{n} + 0}}$$

4. $$\sqrt{\frac{1 + \frac{1}{n}}{\frac{1}{n}}}$$

5. $$\sqrt{\frac{1 + \frac{1}{n}}{\frac{1}{\frac{1}{n}}}}$$

6. $$\sqrt{\frac{\left(1 + \frac{1}{n}\right) \cdot n}{1 \cdot 1}}$$

7. $$\sqrt{\frac{(1) \cdot (n) + \left(\frac{1}{n}\right) \cdot n}{1 \cdot 1}}$$

8. $\sqrt{\frac{n+\frac{n}{n}}{1}}$
9. $\sqrt{\frac{n+\frac{1}{1}}{1}}$
10. $\sqrt{\frac{n+1}{1}}$
11. $\sqrt{n+1}$
12. $\sqrt{17949+1}$
13. $\sqrt{17950}$
14. 133.98

Podemos corroborar este resultado, calculandolo con python (código disponible en el anexo).

Entonces, la proporción que supera la incertidumbre de la predicción a la respuesta media de la calidad es de Resultado: 133.98.

Parte 2: Ecuación de predicción del valor de mercado

b. Ecuación para predecir el valor de mercado del jugador a partir de varias características.
   i. Usando el método de mínimos cuadrados. Explica los indicadores obtenidos (como el coeficiente de determinación y la correlación) y proporciona una breve interpretación de los resultados.
   ii. Usando el método de descenso por gradiente. ¿Son los valores obtenidos iguales a los conseguidos mediante la resolución del sistema de ecuaciones normales? Muestra los resultados obtenidos junto con las últimas iteraciones del algoritmo. Indica los valores de los parámetros utilizados (como tasa de aprendizaje y número de iteraciones).
   iii. Da una interpretación del criterio de corte utilizado en el algoritmo del gradiente. Explica si presenta alguna falla. Si no es una buena condición de corte, ¿puedes sugerir un criterio alternativo más eficaz?

Método de mínimos cuadrados

Para poder obtener una ecuación que permita predecir el valor de mercado de los jugadores en función de todas las variables que sean relevantes, podemos utilizar un script en python, que usa LinearRegression de scikit-learn para aplicar el método de mínimos cuadrados. Una vez ejecutado obtenemos todos los parámetros necesarios:

```
Coeficientes de la regresión (Beta_i):
- β0 = -7706158.26
- β1(age) = -218912.39
- β2(height_cm) = -8577.82
- β3 (weight_kg) = -540.14
- β4(overall) = 245706.94
- β5(potential) = -21977.22
- β6(wage_eur) = 170.29
- β7(international_reputation) = 1171660.84
- β8(weak_foot) = -21193.34
- β9(skill_moves) = -88256.96
Coeficiente de determinación (R^2): 0.74
Suma Total de Cuadrados (STC): 493187887278613056.00
Suma de Cuadrados del Error (SCE): 27687515040464764.00
```

Entonces, la ecuación del hiperplano queda de la siguiente forma:

$$\widehat{y} = \widehat{\beta_0} + \widehat{\beta_1} \cdot x_{age} + \widehat{\beta_2} \cdot x_{hc} + \widehat{\beta_3} \cdot x_{wk} + \widehat{\beta_4} \cdot x_{overall} + \widehat{\beta_5} \cdot x_{potencial} + \widehat{\beta_6} \cdot x_{we} + \widehat{\beta_7} \cdot x_{ir} + \widehat{\beta_8} \cdot x_{wf} + \widehat{\beta_9} \cdot x_{sm}$$

Reemplazando quedaría:

$$\widehat{y} = -7706158.26 + (-218912.39) \cdot x_{age} +$$

$$(-8577.82) \cdot x_{hc} + (-540.14) \cdot x_{wk} + 245706.94 \cdot x_{overall} +$$

$$(-21977.22) \cdot x_{potencial} + 170.29 \cdot x_{we} + 1171660.84 \cdot x_{ir} + (-21193.34) \cdot x_{wf} +$$

$$(-88256.96) \cdot x_{sm}$$

La ecuación obtenida tiene los estimadores $\widehat{\beta_i} \forall i \in [0, 9]$ que serían más apropiados para predecir el valor de mercado de un jugador, en función de todos los parámetros dados y con los datos aportados para cada una de esas variables.

A diferencia del punto anterior, en donde hicimos regresión lineal con cada una de las variables, comparándolas y decidiendo cuál tenía más influencia en la variable independiente, en este punto se intenta obtener el estimador más apropiado para todas las variables, a fin de obtener una función que permita entender cómo todas las variables en su conjunto afectan al valor de mercado de los jugadores.

Además, en relación a los coeficientes obtenidos se puede observar que:

- R^2=0.74 indica que el 74% de la variabilidad del valor de mercado es explicada por las características seleccionadas, lo que sugiere una buena calidad del modelo. Si bien el modelo tiene un buen poder explicativo, aún hay un 26% de la variabilidad que no está capturada por el modelo y podría estar influenciada por las variables que decidimos no incluir desde el dataset.

Método de descenso por gradiente

La función a evaluar es:

$$f\left(x_a, x_h, x_{we}, x_o, x_p, x_{wa}, x_{ir}, x_{wf}, x_{sm}\right)$$

$$= \sum_{i=1}^{n} \left(y_i - \left(b_0 + b_a \cdot x_a + b_h \cdot x_h + b_{we} \cdot x_{we} + b_o \cdot x_o + b_p \cdot x_p + b_{wa} \cdot x_{wa} + b_{ir} \cdot x_{ir} + b_{wf} \cdot x_{wf} + b_{sm} \cdot x_{sm}\right)\right)^2$$

$$= \sum_{i=1}^{n} \left(y_i - b_0 - b_a \cdot x_a - b_h \cdot x_h - b_{we} \cdot x_{we} - b_o \cdot x_o - b_p \cdot x_p - b_{wa} \cdot x_{wa} - b_{ir} \cdot x_{ir} - b_{wf} \cdot x_{wf} - b_{sm} \cdot x_{sm}\right)^2$$

En el apunte de Cálculo en dos o más variables provisto por la cátedra tenemos el **teorema 1.14**: al ser f una sumatoria de polinomios (los cuales a su vez, son funciones continuas en $\Re^9$), f también es una función continua en todo $\Re^9$.

Desarrollando las derivadas parciales de f respecto a cada variable, obtenemos sumatorias de polinomios (las cuales son diferenciables en $\Re^9$). Entonces, siguiendo la **propiedad 2.10** del mismo apunte, cada derivada parcial también es diferenciable en $\Re^9$.

Por lo tanto, es posible hallar un punto mínimo de f a través del método de mínimos cuadrados.

Para ello, convertimos el sistema de ecuaciones normales a forma matricial, buscando que toda la expresión se iguale al vector nulo 0 (Se iguala a cero para poder minimizar la función):

$$X \cdot \beta = Y$$

$$X \cdot \beta - Y = Y - Y$$

$$X \cdot \beta - Y = 0$$

Así, se obtuvo que la función gradiente $\nabla f\left(x_a, x_{hc}, x_{wk}, x_o, x_p, x_{we}, x_{ir}, x_{wf}, x_{sm}\right)$ es $\nabla f(\beta) = X \cdot \beta - Y$.

Ahora de la misma forma pero con un script en python utilizamos el descenso de gradiente para ajustar los parámetros de forma iterativa y así minimizar la función.

Sobre el algoritmo que utilizamos hay algunas consideraciones a tener en cuenta:

- Un parámetro importante del descenso de gradiente es el alpha, denominado como **tasa de aprendizaje**. Si la tasa de aprendizaje es demasiado pequeña, entonces el algoritmo tendrá que pasar por muchas iteraciones para converger, lo que llevará mucho tiempo. Por otro lado, si la tasa de aprendizaje es demasiado alta, es posible que se salte el mínimo global y termine en otro lado, posiblemente incluso más alto que antes. Esto podría hacer que el algoritmo diverja, con valores cada vez mayores, sin encontrar una buena solución.
- **Máximo de iteraciones:** Es importante aclarar, que a pesar de que el algoritmo ejecute el máximo de iteraciones, el resultado arrojado no necesariamente es una buena aproximación de la función f. Por lo tanto es necesario definir un criterio que permita decidir si el resultado obtenido es adecuado o no
- **Criterio de convergencia y tolerancia**: Se calcula la diferencia entre el costo actual y el costo previo del algoritmo. Si esta diferencia es menor que la tolerancia el ciclo se detiene, indicando que se ha alcanzado la convergencia.
- Una última **limitación** es que el descenso de gradiente solo funciona cuando nuestra función es diferenciable en todas partes. En nuestro caso no tenemos esa limitación ya que como probamos al principio del ejercicio nuestra función es diferenciable en todo $\Re^9$.

Dicho esto, y con el script con los parámetros detallados a continuación, se obtienen los siguientes resultados:

- Coeficientes inicializados en cero.
- $\varepsilon = 1e - 6$
- $\eta = 0.01$
- $max\_steps = 10000$

Resultados de los estimadores en las últimas 5 iteraciones del algoritmo:

Iteración -1: [2.22481329e+06 -1.04636113e+06 -5.47098075e+04 1.46775333e+04 1.73499922e+06 -1.33764790e+05 3.24692863e+06 4.54102885e+05 -2.96841433e+02 -2.68464208e+04]

Iteración -2: [2.22481329e+06 -1.04636044e+06 -5.47098226e+04 1.46775693e+04 1.73499829e+06 -1.33764013e+05 3.24692874e+06 4.54102774e+05 -2.96840429e+02 -2.68463381e+04]

Iteración -3: [2.22481329e+06 -1.04635976e+06 -5.47098377e+04 1.46776052e+04 1.73499737e+06 -1.33763234e+05 3.24692884e+06 4.54102663e+05 -2.96839424e+02 -2.68462555e+04]

Iteración -4: [2.22481329e+06 -1.04635908e+06 -5.47098529e+04 1.46776412e+04 1.73499645e+06 -1.33762456e+05 3.24692895e+06 4.54102551e+05 -2.96838419e+02 -2.68461728e+04]

Iteración -5: [2.22481329e+06 -1.04635839e+06 -5.47098680e+04 1.46776773e+04 1.73499552e+06 -1.33761676e+05 3.24692905e+06 4.54102440e+05 -2.96837413e+02 -2.68460900e+04]

Coeficientes óptimos:

- Intercepto: 2224813.29
- age: -1046361.12
- height_cm: -54709.80
- weight_kg: 14677.53
- overall: 1734999.21
- potential: -133764.79
- wage_eur: 3246928.63
- international_reputation: 454102.88
- weak_foot: -296.84
- skill_moves: -26846.42
- Costo final: 3056310441097.4404

Conclusión

El criterio usado es una buena condición de corte, porque se basa en que el vector gradiente tenga una longitud mínima (menor a ε). Esto significa que la tasa de cambio es mínima, lo cual ocurre cuando se está llegando a un punto estacionario. Sin embargo, el costo final(error cuadrático medio entre las predicciones del modelo y los valores reales), que debería ser pequeño si el algoritmo ha funcionado correctamente, nos retornó un valor bastante alto, lo cual indica que se podría hacer algún ajuste adicional para poder obtener estimaciones mejores.

Los estimadores finales tampoco resultaron ser cercanos a los obtenidos con el método de mínimos cuadrados, lo que puede darnos una pauta de que el algoritmo de descenso de gradiente requiera algún ajuste adicional en sus parámetros.

Parte 3: Comportamiento del método de descenso por gradiente

c. **Convergencia del método de descenso por gradiente.** Explicar si el método siempre converge al mínimo de la función. En caso contrario, proporciona un contraejemplo para ilustrar este comportamiento.

El método de descenso por gradiente **no siempre garantiza la convergencia al mínimo global** de una función, y su comportamiento depende de varios factores:

- Si la función tiene múltiples mínimos (locales y globales), el método de descenso por gradiente puede converger a un mínimo local en lugar del mínimo global. Además dependiendo de la función es posible encontrar un punto estacionario o punto silla, el cual no representaría un mínimo de la función.
- Si el tamaño del paso (o tasa de aprendizaje) es muy grande, el algoritmo puede "rebotar" alrededor del mínimo y no converger nunca. Si es demasiado pequeño, el método puede ser muy lento o incluso estancarse.
- El punto de inicio puede afectar si el algoritmo encuentra el mínimo global o local, especialmente en funciones no convexas.
- El descenso por gradiente garantiza la convergencia al mínimo global sólo si la función es convexa. En el caso de una función no convexa, el algoritmo puede quedar atrapado en un mínimo local.

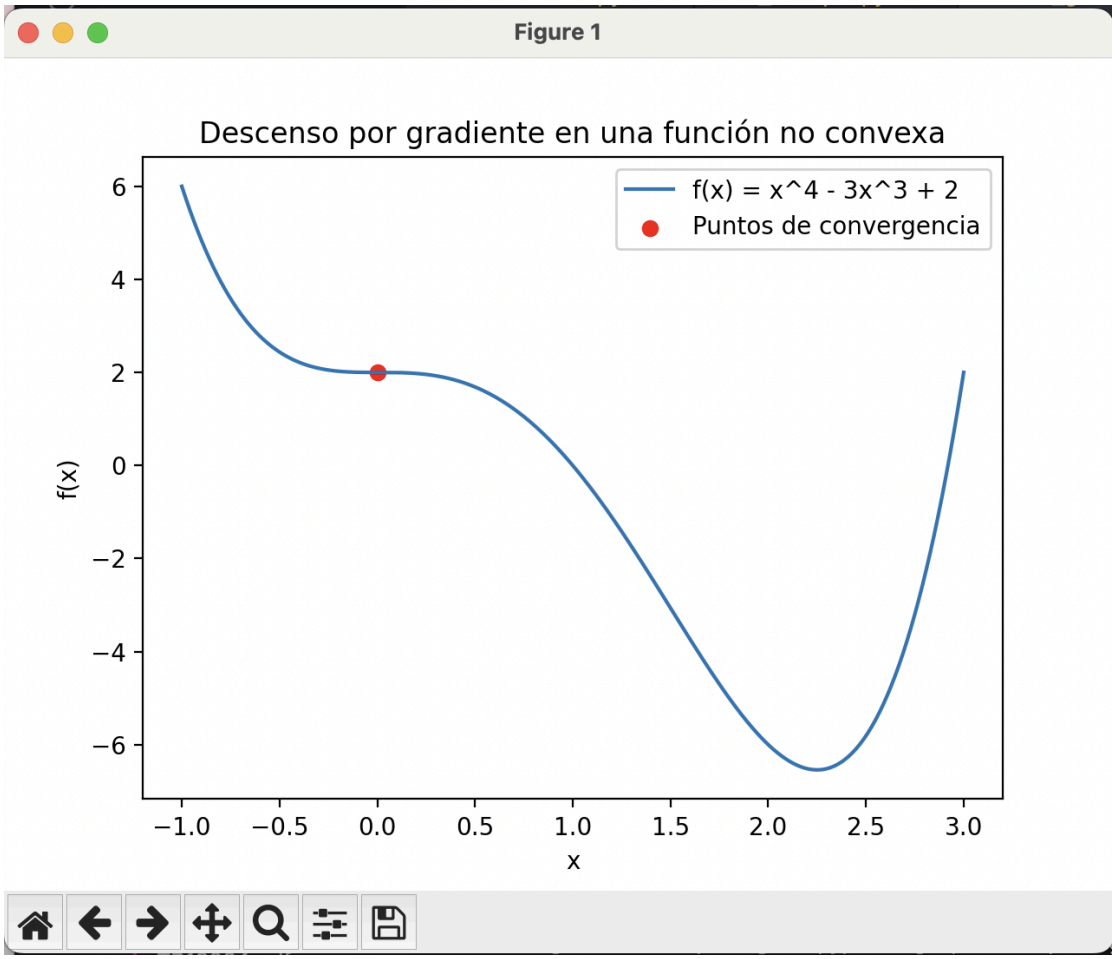
Contraejemplo:

Tenemos por ejemplo la siguiente función:

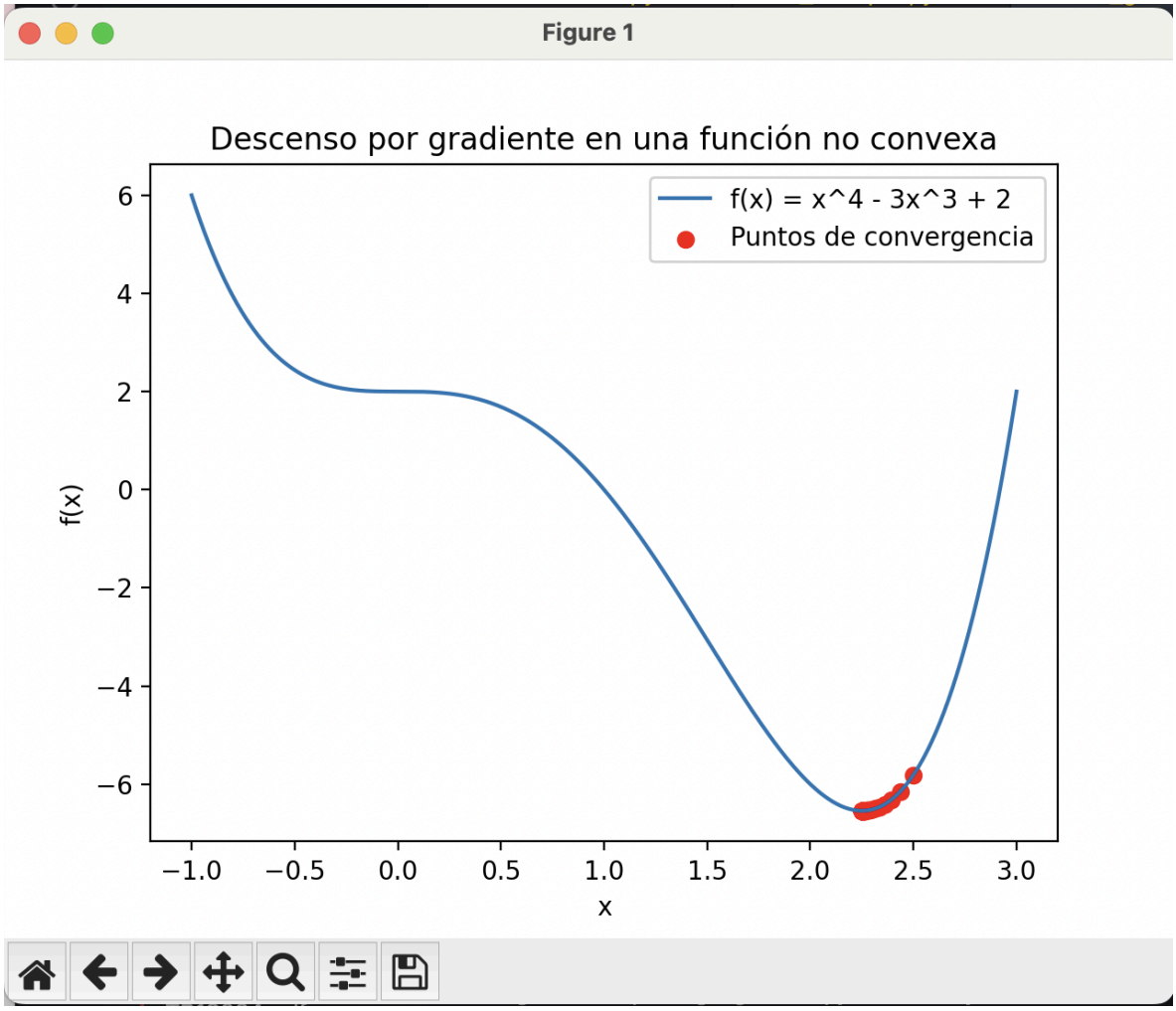
- f(x)=$x^4 - 3x^3 + 2$

Esta función tiene tanto un mínimo local como un mínimo global. Si aplicamos el descenso por gradiente a esta función, dependiendo del punto de inicio y la tasa de aprendizaje, podríamos obtener el mínimo local en vez del mínimo global.

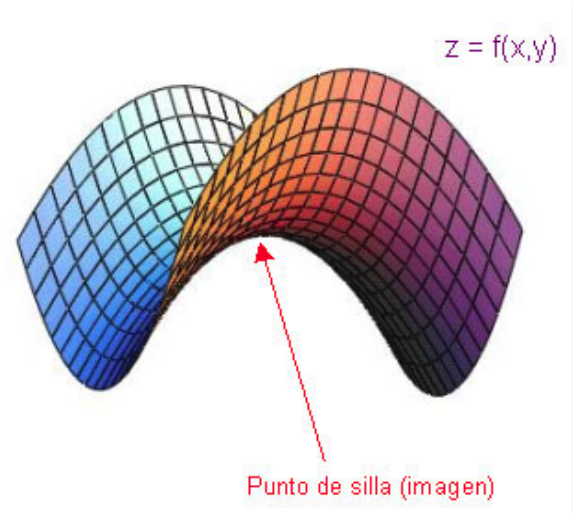
En la siguiente gráfica podemos ver que si empezamos con x=0 el algoritmo se queda estancado en el mínimo local:



En cambio, con un x=2,5 el algoritmo converge al mínimo global efectivamente.



Existen otros ejemplos de funciones que pueden no tener mínimo global como por ejemplo:



Código fuente utilizado:

Código utilizado para el punto a:

```
import pandas as pd
from sklearn.linear_model import LinearRegression
import numpy as np

# Pirmero cargo el archivo de datos
data = pd.read_csv('Path/al/archivo/players_21.csv')

# Luego, selecciono las columnas que me interesan, la columna
value_eur es la que quiero predecir
# y las demas columnas son las caracteristicas que voy a usar para
predecir siempre que no tengan NaN
target = 'value_eur'
features = ['age', 'height_cm', 'weight_kg', 'overall', 'potential',
'wage_eur', 'international_reputation',
            'weak_foot', 'skill_moves']
data = data[[target] + features].dropna()


y = data[target].values.reshape(-1, 1)
# Calculo la media de la columna target
y_mean = np.mean(y)

# Creo un diccionario para almacenar los resultados
results = {}

# Ajusto modelo de regresion lineal simple para cada columna
for feature in features:
    X = data[feature].values.reshape(-1, 1)
    # Entrenar el modelo
    reg = LinearRegression().fit(X, y)
    # Obtener los coeficientes (β1) (β0)
    beta_1 = reg.coef_[0][0]
    beta_0 = reg.intercept_[0]
    # Predicciones del modelo
    y_pred = reg.predict(X)

    # Calculo STC(Suma de los cuadrados totales)
    STC = np.sum((y - y_mean) ** 2)
```

```
    # Calculo SCE(SUma de cuadrados del error) con dos digitos de
decimales
    SCE = np.sum((y - y_pred) ** 2)

    # Guardar los resultados en el diccionario
    results[feature] = {
        'beta_0': beta_0.round(2),
        'beta_1': beta_1.round(2),
        'SCE': SCE,
        'STC': STC
    }
print(type(SCE))
# Mostrar los resultados para cada característica formateados en punto
flotante
for column, result in results.items():
    print(f"Característica: {column}")
    print(f"  β₀ (intercepto): {result['beta_0']}")
    print(f"  β₁ (pendiente): {result['beta_1']}")
    print(f"  SCE (Suma de Cuadrados del Error): {result['SCE']:.2f}")
    print(f"  STC (Suma Total de Cuadrados): {result['STC']:.2f}")
    r = np.sqrt(1 - (SCE / STC))
    r2 = r ** 2
    print(f"  Coeficiente de Correlación Lineal: {r:.2f}")
    print(f"  Coeficiente de Determinación: {r2:.2f}")
    print("-" * 30)

# Guardo los resultados en un archivo csv
results_df = pd.DataFrame(results).T
results_df.to_csv('results.csv')
```

Cálculo de T y p-valor:

```
import numpy as np
from scipy import stats

beta_1 = 218.35  # valor de la pendiente estimada
Sxx = 71891324325l0.45      # Suma de cuadrados de las diferencias de
X
sigma_squared = 79582265271l5.8662  # Estimador insesgado de sigma^2
n = 17949         # Número de observaciones

# Calcular el error estándar de beta_1
SE_beta_1 = np.sqrt(sigma_squared / Sxx)

# Calcular el valor t
```

```
t_stat = beta_1 / SE_beta_1

# Calcular el p-valor (dos colas)
p_val = 2 * stats.t.sf(np.abs(t_stat), df=n-2)

# Mostrar resultados
print(f"t-estadístico: {t_stat:.4f}")
print(f"p-valor: {p_val:.4f}")
```

Cálculo de los intervalos con una confianza del 95%

```
import numpy as np
from scipy import stats

# Datos calculados previamente
beta_0 = 332598.82
beta_1 = 218.35      # Valor de la pendiente estimada
Sxx = 71891324325l0.45          # Suma de los cuadrados de las
diferencias de X
sigma_squared = 79582265271l5.8662 # Estimador insesgado de sigma^2
n = 17949             # Número de observaciones
x_mean = 8853.56        # Media de los valores de X

# Error estándar de beta_1
SE_beta_1 = np.sqrt(sigma_squared / Sxx)

# Error estándar de beta_0
SE_beta_0 = np.sqrt(sigma_squared * (1/n + (x_mean ** 2) / Sxx))

# Valor crítico t para 95% de confianza y n-2 grados de libertad
t_critical = stats.t.ppf(1 - 0.025, df=n-2)

# Intervalo de confianza para beta_1
IC_beta_1_lower = beta_1 - t_critical * SE_beta_1
IC_beta_1_upper = beta_1 + t_critical * SE_beta_1

# Intervalo de confianza para beta_0
IC_beta_0_lower = beta_0 - t_critical * SE_beta_0
IC_beta_0_upper = beta_0 + t_critical * SE_beta_0

# Mostrar resultados
print(f"El valor t_critical es: {t_critical:.4f}")
print(f"Intervalo de confianza al 95% para beta_1:
({IC_beta_1_lower:.2f}, {IC_beta_1_upper:.2f})")
print(f"Intervalo de confianza al 95% para beta_0:
({IC_beta_0_lower:.2f}, {IC_beta_0_upper:.2f})")
```

IP /ICM

```
import math

# Datos calculados previamente
alfa = 0.05
n = 17949
gradoslibertad = n - 2
t_value = 1.9601
Sxx = 71891324325l0.45
sigma_squared = 7958226527145.8662
beta0 = 332598.82
beta1 = 218.1

# Media de la característica
x = 8853.56  # Valor específico de x
x_bar = 8853.56  # Media de la característica fija

# Intervalo de predicción y de confianza para la media
error_prediccion = t_value * math.sqrt(sigma_squared * (1 + (1/n) +
((x - x_bar)**2 / Sxx)))
error_media = t_value * math.sqrt(sigma_squared * ((1/n) + ((x -
x_bar)**2 / Sxx)))

# Proporción entre los dos errores
proporcion = error_prediccion / error_media

# Resultados
print(f"Error de predicción: {error_prediccion}")
print(f"Error de la media: {error_media}")
print(f"Proporción entre error de predicción y error de la media:
{proporcion:.2f}")
```

Regresión Múltiple

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
import numpy as np

# Cargar el archivo CSV
df = pd.read_csv('/path_al_archivo/players_21.csv')
```

```
# Seleccionar las características y la variable objetivo
target = 'value_eur'
selected_features = ['age', 'height_cm', 'weight_kg', 'overall',
'potential', 'wage_eur', 'international_reputation', 'weak_foot',
'skill_moves']

# Crear el conjunto de variables independientes (X) y dependiente (y)
X = df[selected_features]
y = df[target]

# Dividir los datos en conjunto de entrenamiento y prueba
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

# Crear el modelo de regresión lineal
model = LinearRegression()

# Ajustar el modelo con los datos de entrenamiento
model.fit(X_train, y_train)

# Obtener los coeficientes beta_i
coeficientes = model.coef_
intercepto = model.intercept_

# Predecir los valores para el conjunto de prueba
y_pred = model.predict(X_test)

# Calcular el coeficiente de determinación R^2
r2 = r2_score(y_test, y_pred)

# Calcular la matriz de correlación entre las características y el
valor de mercado
correlacion = df[selected_features + [target]].corr()

# Calcular SCE y STC
y_mean = np.mean(y)
STC = np.sum((y - y_mean) ** 2)
SCE = np.sum((y_pred - y_test) ** 2)


# Mostrar los resultados
print("Coeficientes de la regresión (beta_i):")
for feature, coef in zip(selected_features, coeficientes):
    print(f"{feature}: {coef:.2f}")

print(f"\nIntercepto (beta_0): {intercepto:.2f}")
print(f"\nCoeficiente de determinación (R^2): {r2:.2f}")
print(f"\nSuma Total de Cuadrados (STC): {STC:.2f}")
```

```
print(f"Suma de Cuadrados del Error (SCE): {SCE:.2f}")
print("\nMatriz de correlación entre las características y el valor
de mercado:")
print(correlacion)
```

Descenso del Gradiente

```
import numpy as np
import pandas as pd

# Leer datos
data =
pd.read_csv('/Users/camilg/Documents/MATE4/2024/archive/players_21.cs
v')

# Seleccionar las columnas que nos interesan
target = 'value_eur'
features = ['age', 'height_cm', 'weight_kg', 'overall', 'potential',
'wage_eur',
            'international_reputation', 'weak_foot', 'skill_moves']

data = data[[target] + features].dropna()

# Normalizar los datos (es importante para el descenso por gradiente)
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
data_scaled = scaler.fit_transform(data[features])

# Separar la variable dependiente y las independientes
X = np.c_[np.ones(data_scaled.shape[0]), data_scaled]  # Añadir una
columna de 1s para el término independiente (intercepto)
y = data[target].values.reshape(-1, 1)

# Parámetros iniciales
theta = np.zeros((X.shape[1], 1))  # Inicializamos los coeficientes en
0
learning_rate = 0.01  # Tasa de aprendizaje
tolerance = 1e-6  # Tasa de tolerancia para la convergencia
max_iterations = 10000  # Número máximo de iteraciones

# Función de costo (error cuadrático medio)
def compute_cost(X, y, theta):
    m = len(y)
    predictions = X.dot(theta)
    cost = (1 / (2 * m)) * np.sum(np.square(predictions - y))
```

```
    return cost

# Función de descenso por gradiente
def gradient_descent(X, y, theta, learning_rate, tolerance,
max_iterations):
    m = len(y)
    cost_history = []  # Para guardar el historial del costo
    theta_history = []  # Para guardar los valores de theta en cada
iteración

    for iteration in range(max_iterations):
        # Calcular las predicciones
        predictions = X.dot(theta)

        # Calcular el gradiente
        gradient = (1 / m) * X.T.dot(predictions - y)

        # Actualizar los valores de theta
        theta = theta - learning_rate * gradient

        # Guardar el costo en cada iteración
        cost = compute_cost(X, y, theta)
        cost_history.append(cost)

        # Guardar los valores de theta para cada iteración
        theta_history.append(theta.copy())

        # Verificar si la mejora es lo suficientemente pequeña para
detenerse (criterio de convergencia)
        if iteration > 0 and np.abs(cost_history[-2] -
cost_history[-1]) < tolerance:
            print(f"Convergencia alcanzada en la iteración
{iteration}")
            break

    return theta, cost_history, theta_history

# Ejecutar el descenso por gradiente
theta_optimal, cost_history, theta_history = gradient_descent(X, y,
theta, learning_rate, tolerance, max_iterations)

# Mostrar los coeficientes óptimos obtenidos
print("Coeficientes óptimos (theta):")
for i, feature in enumerate(['Intercepto'] + features):
    print(f"{feature}: {theta_optimal[i][0]}")

# Mostrar el costo final
print(f"Costo final: {cost_history[-1]}")
```

```
# Mostrar los valores de theta en las últimas 5 iteraciones
print("\nValores de theta durante las últimas 5 iteraciones:")
for i in range(1, 6):
    print(f"Iteración {-i}: {theta_history[-i].flatten()}")
```

Referencias utilizadas

Regresión lineal múltiple. StatDeveloper

Regresión Lineal con Gradiente Descendente | by BackStreetCode | Medium

Algoritmo del Gradiente

Regresión Lineal Múltiple con NumPy | Gradient Descent | Boston House Dataset

Regresión lineal con gradiente descendente en Python – Symmetric

Descenso de gradiente (artículo) | Khan Academy

Regresión múltiple

Fórmulas básicas en la regresión lineal simple