Fifa practice

https://github.com/Florinx28/PracticaFifa

El objetivo de esta práctica es predecir el valor de los jugadores a partir de la información de estos.

Importar Librerias

Primero, vamos a importar la librerias que utilizaremos.

In [ ]:

```
import os

from sklearn.model_selection import train_test_split
from sklearn import linear_model
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score

import pandas as pd
import numpy as np
```

Leer los datos

Para la lectura y analisis de los datos utilizaremos `pandas` y los servicios de la libreria `os` .

In [ ]:

```
df = pd.read_csv(os.path.join("..", "in", "/content/fifa.csv"))
df
```

Out[ ]:

	Unnamed: 0	ID	Name	Age	Photo	Nationality	Flag	Overall	P
0	0	158023	L. Messi	31	https://cdn.sofifa.org/players/4/19/158023.png	Argentina	https://cdn.sofifa.org/flags/52.png	94	
1	1	20801	Cristiano Ronaldo	33	https://cdn.sofifa.org/players/4/19/20801.png	Portugal	https://cdn.sofifa.org/flags/38.png	94	
2	2	190871	Neymar Jr	26	https://cdn.sofifa.org/players/4/19/190871.png	Brazil	https://cdn.sofifa.org/flags/54.png	92	
3	3	193080	De Gea	27	https://cdn.sofifa.org/players/4/19/193080.png	Spain	https://cdn.sofifa.org/flags/45.png	91	
4	4	192985	K. De Bruyne	27	https://cdn.sofifa.org/players/4/19/192985.png	Belgium	https://cdn.sofifa.org/flags/7.png	91	
...	...	...	...	...	...	...	...	...	
18202	18202	238813	J. Lundstram	19	https://cdn.sofifa.org/players/4/19/238813.png	England	https://cdn.sofifa.org/flags/14.png	47	
18203	18203	243165	N. Christoffersson	19	https://cdn.sofifa.org/players/4/19/243165.png	Sweden	https://cdn.sofifa.org/flags/46.png	47	
18204	18204	241638	B. Worman	16	https://cdn.sofifa.org/players/4/19/241638.png	England	https://cdn.sofifa.org/flags/14.png	47	
18205	18205	246268	D. Walker-Rice	17	https://cdn.sofifa.org/players/4/19/246268.png	England	https://cdn.sofifa.org/flags/14.png	47	
18206	18206	246269	G. Nugent	16	https://cdn.sofifa.org/players/4/19/246269.png	England	https://cdn.sofifa.org/flags/14.png	46	

18207 rows × 89 columns

Ahora seleccionaremos las columnas. El resultado será un objeto de la libreria `pandas` , un objeto `series` . Este objeto se puede convertir en un array estándar.

In [ ]:

```
name = df["Name"]
name
```

Out[ ]:

```
0                  L. Messi
1          Cristiano Ronaldo
2                  Neymar Jr
3                     De Gea
4               K. De Bruyne
                   ...
18202           J. Lundstram
18203    N. Christoffersson
18204              B. Worman
18205         D. Walker-Rice
18206              G. Nugent
Name: Name, Length: 18207, dtype: object
```

Analisis de datos

Primero, queremos saber como son los datos. Para ello aumentaremos el número de columnas que se van a mostrar. Posteriormente, utilizaremos instrucciones especiales que nos mostrará como se estructurán los datos.

In [ ]:

```
pd.set_option('display.max_columns', None)
```

In [ ]:

```
df.head()
```

Out[ ]:

	Unnamed: 0	ID	Name	Age	Photo	Nationality	Flag	Overall	Potential	
0	0	158023	L. Messi	31	https://cdn.sofifa.org/players/4/19/158023.png	Argentina	https://cdn.sofifa.org/flags/52.png	94	94	
1	1	20801	Cristiano Ronaldo	33	https://cdn.sofifa.org/players/4/19/20801.png	Portugal	https://cdn.sofifa.org/flags/38.png	94	94	
2	2	190871	Neymar Jr	26	https://cdn.sofifa.org/players/4/19/190871.png	Brazil	https://cdn.sofifa.org/flags/54.png	92	93	F
3	3	193080	De Gea	27	https://cdn.sofifa.org/players/4/19/193080.png	Spain	https://cdn.sofifa.org/flags/45.png	91	93	M
4	4	192985	K. De Bruyne	27	https://cdn.sofifa.org/players/4/19/192985.png	Belgium	https://cdn.sofifa.org/flags/7.png	91	92	M

In [ ]:

```
df.describe()
```

Out[ ]:

	Unnamed: 0	ID	Age	Overall	Potential	Special	International Reputation	Weak Foot	Skill Moves	
count	18207.000000	18207.000000	18207.000000	18207.000000	18207.000000	18207.000000	18159.000000	18159.000000	18159.000000	1
mean	9103.000000	214298.338606	25.122206	66.238699	71.307299	1597.809908	1.113222	2.947299	2.361308	
std	5256.052511	29965.244204	4.669943	6.908930	6.136496	272.586016	0.394031	0.660456	0.756164	
min	0.000000	16.000000	16.000000	46.000000	48.000000	731.000000	1.000000	1.000000	1.000000	
25%	4551.500000	200315.500000	21.000000	62.000000	67.000000	1457.000000	1.000000	3.000000	2.000000	
50%	9103.000000	221759.000000	25.000000	66.000000	71.000000	1635.000000	1.000000	3.000000	2.000000	
75%	13654.500000	236529.500000	28.000000	71.000000	75.000000	1787.000000	1.000000	3.000000	3.000000	
max	18206.000000	246620.000000	45.000000	94.000000	95.000000	2346.000000	5.000000	5.000000	5.000000	

Los datos contienen *NaN* (Not a Number). Por lo que también hay que tratarlos.

Tratamiento de NaNs

Qué NaNs tenemos? Mediante la siguiente instrucción veremos todas las columnas que contienen al menos un NaN. Como podremos ver, serán la mayoría.

In [ ]:

```
df.columns[df.isna().any()].tolist()
```

Out[ ]:

```
['Club',
 'Preferred Foot',
 'International Reputation',
 'Weak Foot',
 'Skill Moves',
 'Work Rate',
 'Body Type',
 'Real Face',
 'Position',
 'Jersey Number',
 'Joined',
 'Loaned From',
 'Contract Valid Until',
 'Height',
 'Weight',
 'LS',
 'ST',
 'RS',
 'LW',
 'LF',
 'CF',
 'RF',
 'RW',
 'LAM',
 'CAM',
 'RAM',
 'LM',
 'LCM',
 'CM',
 'RCM',
 'RM',
 'LWB',
 'LDM',
 'CDM',
 'RDM',
 'RWB',
 'LB',
 'LCB',
 'CB',
 'RCB',
 'RB',
 'Crossing',
 'Finishing',
 'HeadingAccuracy',
 'ShortPassing',
 'Volleys',
 'Dribbling',
 'Curve',
 'FKAccuracy',
 'LongPassing',
 'BallControl',
 'Acceleration',
 'SprintSpeed',
 'Agility',
 'Reactions',
 'Balance',
 'ShotPower',
 'Jumping',
 'Stamina',
 'Strength',
 'LongShots',
 'Aggression',
 'Interceptions',
 'Positioning',
 'Vision',
 'Penalties',
```

```
 'Composure',
 'Marking',
 'StandingTackle',
 'SlidingTackle',
 'GKDiving',
 'GKHandling',
 'GKKicking',
 'GKPositioning',
 'GKReflexes',
 'Release Clause']
```

Podemos ver que casi todos los jugadores tienen un NaN. Hay muchas columnas que mayoritariamente son NaN como por ejemplo **Loaned From**. Posteriormente veremos como tratarlos, en algunos casos habrá que transformarlos en un valor númerico de alguna forma y en otros, se podrán eliminar debido a que serán un procentaje bajo del dataset los cuales los tengan.

La columna **Values**, no tiene ningun NaN pero no es un número. Como cambiamos el formato?

Primero definimos una función que pase de un formato a otro:

In [ ]:

```
def value_to_float(x):
    """
    From K and M to float.

    """
    x = x.replace('€', '')
    ret_val = 0.0

    if type(x) == float or type(x) == int:
        ret_val = x
    if 'K' in x:
        if len(x) > 1:
            ret_val = float(x.replace('K', ''))
        ret_val = ret_val *1000
    if 'M' in x:
        if len(x) > 1:
            ret_val = float(x.replace('M', ''))
        ret_val = ret_val * 1000000.0
    return ret_val
```

Ahora aplicamos la función a la columna. Esta función la tendremos que aplicar en un futuro a más columnas que sufren el mismo problema que esta.

In [ ]:

```
df["Value"] = df["Value"].apply(value_to_float)
```

Eliminación de columnas

En primer lugar vamos a eliminar las columnas que en mi opinión no aporta valores significativos para realizar la predicción.

In [ ]:

```
df = df.drop(columns = ["ID","Name","Photo","Flag","Club Logo","Real Face","Joined","Special","Preferred
"LWB","LDM","CDM","RDM","RWB","LB","LCB","CB","RCB","RB"],axis=1)
```

Conjunto de columnas eliminadas:

1. **ID:** Esta columna contiene el identificador del jugador. No tiene relación con el precio de este.
2. **Name:** Esta columna contiene el nombre del jugador. No creo que el nombre de un jugador cualquiera haga que valga mas que otro, por lo cual la elimino.
3. **Photo:** Esta columna contiene un link a la imagen del jugador. Independientemente del tipo de dato que es, el aspecto del jugador tampoco influye en el precio.
4. **Flag:** Esta columna contiene un link a la imagen de la bandera del jugador. Por lo que al igual que en el caso anterior, no influye.
5. **Club Logo:** Esta columna contiene un link a la imagen del logo del club al cual pertenece el jugador. Independientemente de como sea el logo, no altera el precio del jugador
6. **Real Face:** Independientemente de si su cara es la real o no, no modifica el precio final del jugador.
7. **Joined:** Contiene la fecha en la que un jugador se unió al club, la cual no debería esta relacionada con el precio del jugador.
8. **Special:** Realmente no estoy muy seguro de lo que indica esta variable, por lo que mejor la saco del dataset.
9. **Preferred Foot:** Realmente no creo que influya en el precio si el jugador es zurdo o diestro, sino que tan bueno es con su pierna mala **Weak Foot**.
10. **Contract Valid Until:** El año en el cual se termina el contrato no es influyente, es parecido al caso de **Joined**.
11. **Conjunto de elementos desde LS hasta RB:** Este conjunto de datos hace referencia al posible cambio de un jugador de una posición a otra mediante cartas de entrenamiento del juego. No lo incluyo en el conjunto de datos porque no los veo como atributos propios del jugador, sino como unos valores que FIFA ha asignado a los jugadores dependiendo de su posición y mediante los cuales los usuarios tienen una mayor facilidad para elaborar sus equipos con los jugadores que quieren.

Preparación de los datos

Una vez establecidos los datos que vamos a usar, ahora hay que preparalos para que el algoritmo de predicción los pueda usar. Para eso, tendremos que eliminar ciertos signos como "M" o "K", adaptar ciertos valores como la nacionalidad al igual que los equipos y también eliminar todos los valores NaN.

Primero vamos a proceder con la eliminación de los NaN, si nos fijamos en la columna de "Loaned From"(cedidos) la mayoria son NaNs.

In [ ]:

```
df["Loaned From"].isna().sum()
```

Out[ ]:

```
16943
```

Lo primero que haremos, será sustituir todos los NaNs por zero.

In [ ]:

```
df["Loaned From"].fillna(0,inplace = True)
```

Una vez establecidos a zero, los valores de la columan "Loaned From" serán 0 o el nombre del equipo del cual están cedidos, es decir su equipo original.

In [ ]:

```
df["Loaned From"].value_counts()
```

Out[ ]:

```
0                                16943
Atalanta                            20
Sassuolo                            18
Juventus                            17
SL Benfica                          17
                                 ...
LASK Linz                            1
SV Sandhausen                        1
La Berrichonne de Châteauroux        1
Vitória Guimarães                    1
CD O'Higgins                         1
Name: Loaned From, Length: 342, dtype: int64
```

A continuación, sustituiremos los valores de la columna "Club" por los valores de "Loaned From" que sean diferentes de cero. De esta forma, a los jugadores cedidos les saldrá en la columna "Club" su equipo original y no al que han sido cedidos.

In [ ]:

```
df['Club'] = np.where(df['Loaned From'] != 0, df['Loaned From'], df["Club"])
```

A los jugadores cedidos les pondremos el valor de 1 en lugar del nombre de su equipo, y los jugadores no cedidos seguirán con el valor de cero.

In [ ]:

```
df['Loaned From'] = np.where(df['Loaned From'] != 0, 1, df["Loaned From"])
```

In [ ]:

```
df["Loaned From"].value_counts()
df.head()
```

Out[ ]:

	Unnamed: 0	Age	Nationality	Overall	Potential	Club	Value	Wage	International Reputation	Weak Foot	Skill Moves	Work Rate	Body Type	Position	Je Nun
0	0	31	Argentina	94	94	FC Barcelona	110500000.0	€565K	5.0	4.0	4.0	Medium/ Medium	Messi	RF	
1	1	33	Portugal	94	94	Juventus	77000000.0	€405K	5.0	4.0	5.0	High/ Low	C. Ronaldo	ST	
2	2	26	Brazil	92	93	Paris Saint-Germain	118500000.0	€290K	5.0	5.0	5.0	High/ Medium	Neymar	LW	
3	3	27	Spain	91	93	Manchester United	72000000.0	€260K	4.0	3.0	1.0	Medium/ Medium	Lean	GK	
4	4	27	Belgium	91	92	Manchester City	102000000.0	€355K	4.0	5.0	4.0	High/ High	Normal	RCM	

Ahora el problema esta con la columna **Club**, que también contiene Nans.

In [ ]:

```
df.loc[df.Club != df.Club]
```

Out[ ]:

	Unnamed: 0	Age	Nationality	Overall	Potential	Club	Value	Wage	International Reputation	Weak Foot	Skill Moves	Work Rate	Body Type	Position	Jersey Number	Loar Fr
452	452	24	Argentina	80	85	NaN	0.0	€0	2.0	4.0	4.0	Medium/ Medium	Normal	CM	5.0	
538	538	33	Sweden	80	80	NaN	0.0	€0	2.0	4.0	2.0	High/ Medium	Normal	LCB	4.0	
568	568	26	Russia	79	81	NaN	0.0	€0	1.0	3.0	1.0	Medium/ Medium	Normal	GK	12.0	
677	677	29	Russia	79	79	NaN	0.0	€0	2.0	3.0	3.0	High/ High	Lean	RB	2.0	
874	874	29	Russia	78	78	NaN	0.0	€0	2.0	3.0	3.0	High/ Medium	Stocky	ST	22.0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
17197	17197	21	India	55	64	NaN	0.0	€0	1.0	2.0	1.0	Medium/ Medium	Normal	GK	1.0	
17215	17215	26	Finland	55	57	NaN	0.0	€0	1.0	3.0	2.0	Medium/ High	Normal	RB	3.0	
17339	17339	23	India	54	63	NaN	0.0	€0	1.0	3.0	2.0	Medium/ Low	Normal	NaN	NaN	
17436	17436	20	India	54	67	NaN	0.0	€0	1.0	3.0	2.0	Medium/ Medium	Normal	NaN	NaN	
17539	17539	21	India	53	62	NaN	0.0	€0	1.0	3.0	2.0	High/ Medium	Lean	NaN	NaN	

241 rows × 53 columns

El problema es que hay jugadores que su equipo no esta en el FIFA, como este conjunto es muy pequeño, eliminarlo no supone un problema para la predicción.

In [ ]:

```
df = df.dropna(subset = ['Club'])
```

A pesar de haber eliminado los NANs, aún tenemos un problema. ¿Como usamos estos datos para el algoritmo? Bueno, para su uso podemos utilizar el proceso **One Hot Enconding** el cual convierte las variables categóricas en una representación binaria de estas, de tal forma que la función de predicción si puede tomar estos valores y al mismo tiempo no se mal entienden los valores.

In [ ]:

```
clb = df.pop("Club")

df = pd.concat([df.reset_index(drop=True), pd.get_dummies(clb, prefix='clb').reset_index(drop=True)], axi
```

Una vez tratado el club,repetimos el proceso columnas posición y nacionalidad debido a que presentan el mismo problema. Aunque antes eliminamos los NaNs en caso de que haya.

In [ ]:

```
df["Position"].isna().sum()
```

Out[ ]:

```
48
```

In [ ]:

```
df = df.dropna(subset = ['Position'])
```

In [ ]:

```
Pos = df.pop("Position")
df = pd.concat([df.reset_index(drop=True), pd.get_dummies(Pos, prefix='Pos').reset_index(drop=True)], axi
```

In [ ]:

```
df["Nationality"].isna().sum()
```

Out[ ]:

```
0
```

In [ ]:

```
Nat = df.pop("Nationality")
df = pd.concat([df.reset_index(drop=True), pd.get_dummies(Nat, prefix='Nat').reset_index(drop=True)], axi
```

La columna "Work Rate" hace referencia al comportamiento del jugador en términos de ataque y defensa.Primero, vamos a eliminar los posibles NaNs de esta columna.

In [ ]:

```
df["Work Rate"].isna().sum()
```

Out[ ]:

```
0
```

In [ ]:

```
df = df.dropna(subset = ['Work Rate'])
```

A continuación observamos los posibles valores de esta columna. Como podemos ver se divide en dos partes, la primera, que hace referencia al comportamiento en ataque y la segunda al comportamiento en defensa.

In [ ]:

```
df["Work Rate"].value_counts()
```

Out[ ]:

```
Medium/ Medium    9685
High/ Medium      3131
Medium/ High      1660
High/ High        1007
Medium/ Low        840
High/ Low          686
Low/ Medium        440
Low/ High          435
Low/ Low            34
Name: Work Rate, dtype: int64
```

Con esos valores, podemos volver a aplicar la técnica **One Hot Enconding**.

In [ ]:

```
Wr = df.pop("Work Rate")
df = pd.concat([df.reset_index(drop=True), pd.get_dummies(Wr, prefix='Wr').reset_index(drop=True)], axis=
```

Ahora tenemos un problema con el "body type" al igual que las columnas tratadas anteriormente son variables de tipo String que no nos sirven para realizar la predicción.

In [ ]:

In [ ]:

```
df["Body Type"].isna().sum()
```

Out[ ]:

```
0
```

In [ ]:

```
df["Body Type"].value_counts()
```

Out[ ]:

```
Normal                10436
Lean                   6351
Stocky                 1124
Akinfenwa                 1
Shaqiri                   1
Neymar                    1
C. Ronaldo                1
Courtois                  1
PLAYER_BODY_TYPE_25       1
Messi                     1
Name: Body Type, dtype: int64
```

Como vemos, hay resultados de cuerpos únicos como Messi, C. Ronaldo... Estos cuerpos mejor añadirlos a la mayoría, en este caso Normal

In [ ]:

```
def AgruparCuerpos(x):
  if x == "Normal" or x == "Lean" or x == "Stocky":
    return x
  return "Normal"
```

In [ ]:

```
df["Body Type"] = df["Body Type"].apply(AgruparCuerpos)
```

In [ ]:

```
df["Body Type"].value_counts()
```

Out[ ]:

```
Normal    10443
Lean       6351
Stocky     1124
Name: Body Type, dtype: int64
```

Una vez tratado esos valores extremos, podemos volver a aplicar la técnica **One Hot Encoding** para los valores de esta columna.

In [ ]:

```
Body = df.pop("Body Type")
df = pd.concat([df.reset_index(drop=True), pd.get_dummies(Body, prefix='Body').reset_index(drop=True)], a
```

In [ ]:

```
df.head()
```

Out[ ]:

	Unnamed: 0	Age	Overall	Potential	Value	Wage	International Reputation	Weak Foot	Skill Moves	Jersey Number	Loaned From	Height	Weight	Crossing	Finishing	H
0	0	31	94	94	110500000.0	€565K	5.0	4.0	4.0	10.0	0	5'7	159lbs	84.0	95.0	
1	1	33	94	94	77000000.0	€405K	5.0	4.0	5.0	7.0	0	6'2	183lbs	84.0	94.0	
2	2	26	92	93	118500000.0	€290K	5.0	5.0	5.0	10.0	0	5'9	150lbs	79.0	87.0	
3	3	27	91	93	72000000.0	€260K	4.0	3.0	1.0	1.0	0	6'4	168lbs	17.0	13.0	
4	4	27	91	92	102000000.0	€355K	4.0	5.0	4.0	7.0	0	5'11	154lbs	93.0	82.0	

A continuación trataremos el conjunto de columnas que tengan símbolos en sus valores. En primer lugar, empezaremos con la columna **Wage** para quitarle los simbolos "€" y "K". Para ello aplicamos la función definida anteriormente. Aunque primero miraremos si contiene algún NaN.

In [ ]:

```
df["Wage"].isna().sum()
df["Wage"] = df["Wage"].apply(value_to_float)
```

Ahora los problemas están en la alutra y el peso, debido a que uno esta en pies y pulgadas y el otro en libras. Podemos convertir ambos en centímetro i kilos.

In [ ]:

```
df["Height"].isna().sum()
```

Out[ ]:

```
0
```

In [ ]:

```
def Altura(x):
  x=x.split("'")
  Pie = float(x[0]) * 30.48
  Pulgada = float(x[1]) * 2.54
  return Pie + Pulgada
```

In [ ]:

```
df["Height"] = df["Height"].apply(Altura)
```

In [ ]:

```
df["Weight"].isna().sum()
```

Out[ ]:

```
0
```

In [ ]:

```
def Libras(x):
  x=x.rstrip("lbs")
  return float(x)
```

In [ ]:

```
df["Weight"] = df["Weight"].apply(Libras)
```

Finalmente, hay que tratar la columna de "Release Clause". En este caso no podemos eliminar los NaNs debido a que es un valor muy importante por lo que podriamos subsituirlos por zeros y así simular que no tiene clausura de liberación.

In [ ]:

```
df["Release Clause"].isna().sum()
```

Out[ ]:

```
1275
```

In [ ]:

```
df["Release Clause"].fillna("€0",inplace = True)
```

In [ ]:

```
df["Release Clause"] = df["Release Clause"].apply(value_to_float)
```

In [ ]:

```
df.head()
```

Out[ ]:

	Unnamed: 0	Age	Overall	Potential	Value	Wage	International Reputation	Weak Foot	Skill Moves	Jersey Number	Loaned From	Height	Weight	Crossing	Finishing
0	0	31	94	94	110500000.0	565000.0	5.0	4.0	4.0	10.0	0	170.18	159.0	84.0	95.0
1	1	33	94	94	77000000.0	405000.0	5.0	4.0	5.0	7.0	0	187.96	183.0	84.0	94.0
2	2	26	92	93	118500000.0	290000.0	5.0	5.0	5.0	10.0	0	175.26	150.0	79.0	87.0
3	3	27	91	93	72000000.0	260000.0	4.0	3.0	1.0	1.0	0	193.04	168.0	17.0	13.0
4	4	27	91	92	102000000.0	355000.0	4.0	5.0	4.0	7.0	0	180.34	154.0	93.0	82.0

Predicción

Finalmente vamos a intentar predecir el valor con todas las columnas arregladas.

In [ ]:

```
val = df.pop("Value")
```

Ahora que ya tenemos todos los datos preparados y en su correspondiente formato ya podemos empezar a entrenar el modelo. Pero antes tenemos que divir los datos en el subset de testeo y en el de entrenamiento para que así nuestro modelo pueda generalizar de un forma correcta, de tal forma que cuando reciba un dato que no ha estado en subset de entrenamiento pueda calcular su valor con el mínimo error posible. En nuestro caso lo dividiremos de la siguiente forma:

- **Subset de testeo** : 33% de los datos
- **Subset de entrenamiento**: 67% de los daots

In [ ]:

```
X_train, X_test, y_train, y_test = train_test_split(df, val, test_size=0.33, random_state=42)
```

In [ ]:

```
len(X_train)
```

Out[ ]:

```
12005
```

Procedemos a entrenar el modelo de regresión lineal.

In [ ]:

```
reg = linear_model.LinearRegression().fit(X_train, y_train)
```

Finalmente haremos uso dela métrica R^2 para la regresión, utilizamos la implementación desde sickit-learn. R^2 es la proporción de varianza de la variable dependiente la cual es predecible mediante las variables independientes, es decir, es la varianza entre la variable que intentamos predecir el valor del jugador y sus otras variables. El valor resultante esta comprendido entre 0 y 1 (aunque a veces también puede ser negativo), cuanto mayor sea este resultado, tendremos un mejor ajuste entre la predicción y el valor real del jugador.

In [ ]:

```
preds = reg.predict(X_test)
```

In [ ]:

```
preds[0]
```

Out[ ]:

```
-291195.25978500955
```

In [ ]:

```
y_test[0]
```

Out[ ]:

```
110500000.0
```

In [ ]:

```
r2_score(preds, y_test)
```

Out[ ]:

```
0.9703269058031849
```