```
Práctica ‘4’ {

    [Regresión Polinomial]

        < Profesora: Consuelo Varinia García Mendoza >
        < Alumna: Vianey Maravilla Pérez

}
```

00

[Introducción]

La Práctica que se realizó en esta ocasión se resume a encontrar la mejor manera de entrenar un modelo utilizando alguno de los dos tipos de regresión (lineal o polinomial).
Previamente se llevo a cabo el uso de datos sin escalar y sobre todo se utilizó como primera instancia la regresión polinomial mediante gradiente descendiente estocástico con polinomios de grado 2 y 3.

00

[Regresión Lineal]

La regresión lineal es un método estadístico que trata de modelar la relación entre una variable continua y una o más variables independientes mediante el ajuste de una ecuación lineal.
Se le llama regresión lineal simple cuando solo hay una variable independiente y regresión lineal múltiple cuando hay más de una.
Dependiendo del contexto, a la variable modelada se le conoce como variable dependiente o variable respuesta, y a las variables independientes como regresores, features

01

{

[Código Fuente]

}

Código Fuente

Importación de las librerías que se ocuparan a lo largo de la programación de la regresión polinomial

In [1]:

```
# Importación de librerias
import pandas as pd
import numpy as np
import operator
import os
from sklearn import preprocessing
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import SGDRegressor
from sklearn.model_selection import KFold
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
from sklearn.metrics import r2_score
from sklearn.preprocessing import PolynomialFeatures
import matplotlib.pyplot as plt
```

Código Fuente

Creación del dataset “cal_housing.csv” en un dataframe para poder trabajar de mejor manera con él

In [2]:
```
# Creación del archivo a trabajar en un dataframe
dataframe = pd.read_csv('./cal_housing.csv',sep = ',', engine = 'python')
```

Código Fuente

Mediante el dataframe ya creado hacemos la debida identificación del corpus y las etiquetas

In [3]:

```
# Corpus

X = dataframe.drop('medianHouseValue',axis = 1).values

# Etiquetas

y = dataframe['medianHouseValue'].values
```

Código Fuente

Creación del conjunto de entrenamiento y el conjunto de prueba

In [4]:

```
# Generación Data Test y Data Train

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 0)
```

Código Fuente

Creamos una clase validación

In [5]:

```
class validation_set:

    def __init__(self, X_train, y_train, X_test, y_test):
        self.X_train = X_train
        self.y_train = y_train
        self.X_test = X_test
        self.y_test = y_test

############################################ VALIDATION SETS ################################################################
validation_sets = []
kf = KFold(n_splits = 10)
for train_index, test_index in kf.split(X_train):
    X_train_v, X_test_v = X_train[train_index], X_train[test_index]
    y_train_v, y_test_v = y_train[train_index], y_train[test_index]
    validation_sets.append(validation_set(X_train_v, y_train_v, X_test_v, y_test_v))

############################################ CREACIÓN DE LIST ################################################################

regression_type_list = list()
scale_type_list = list()
learning_rate_list = list()
eta0_list = list()
iterations_list = list()
mse_list = list()
r2_list = list()
```

Código Fuente

Creamos la “clase” o más bien definimos una variable para la regresión lineal

In [6]:

```
# Definición de la regresión Lineal

def linear_regression(X_train, X_test, y_train, y_test, eta0 = 0.001, learning_rate = 'constant', iterations = 10000, standa
    if standar_scale:
        X_train = preprocessing.StandardScaler().fit_transform(X_train)
        X_test = preprocessing.StandardScaler().fit_transform(X_test)
    if robust_scale:
        X_train = preprocessing.RobustScaler().fit_transform(X_train)
        X_test = preprocessing.RobustScaler().fit_transform(X_test)
    regr = SGDRegressor(learning_rate = learning_rate, eta0 = eta0, max_iter = iterations)
    regr.fit(X_train, y_train)
    y_test_pred = regr.predict(X_test)
    mse = mean_squared_error(y_test, y_test_pred)
    r2 = r2_score(y_test, y_test_pred)
    return [mse, r2]
```

Código Fuente

Creamos la “clase” o más bien definimos una variable para la regresión polinomial

In [7]:

```
# Definición de la regresión Polinomial

def polynomial_regression(X_train, X_test, y_train, y_test, degree = 2, eta0 = 0.001, learning_rate = 'constant', iterations
    polynomial_features= PolynomialFeatures(degree = degree)
    X_train = polynomial_features.fit_transform(X_train)
    X_test = polynomial_features.fit_transform(X_test)
    if standar_scale:
        X_train = preprocessing.StandardScaler().fit_transform(X_train)
        X_test = preprocessing.StandardScaler().fit_transform(X_test)
    if robust_scale:
        X_train = preprocessing.RobustScaler().fit_transform(X_train)
        X_test = preprocessing.RobustScaler().fit_transform(X_test)
    regr = SGDRegressor(learning_rate = learning_rate, eta0 = eta0, max_iter = iterations)
    regr.fit(X_train, y_train)
    y_test_pred = regr.predict(X_test)
    mse = mean_squared_error(y_test, y_test_pred)
    r2 = r2_score(y_test, y_test_pred)
    return [mse, r2]
```

Código Fuente

In [8]:

```python
# Definición para obtener los mejores parametros

def get_best_params(validation_sets, eta0, learnin_rate, iterations, type_regression, scale_type):
    c = 1
    mses = list()
    r2s = list()
    n = len(validation_sets)
    for validation_set in validation_sets:
        if scale_type == 'none':
            if type_regression == 1:
                mse, r2 = linear_regression(validation_set.X_train, validation_set.X_test, validation_set.y_train, validatio
            else:
                mse, r2 = polynomial_regression(validation_set.X_train, validation_set.X_test, validation_set.y_train, valid
        elif scale_type == 'std':
            if type_regression == 1:
                mse, r2 = linear_regression(validation_set.X_train, validation_set.X_test, validation_set.y_train, validatio
            else:
                mse, r2 = polynomial_regression(validation_set.X_train, validation_set.X_test, validation_set.y_train, valid
        elif scale_type == 'robust':
            if type_regression == 1:
                mse, r2 = linear_regression(validation_set.X_train, validation_set.X_test, validation_set.y_train, validatio
            else:
                mse, r2 = polynomial_regression(validation_set.X_train, validation_set.X_test, validation_set.y_train, valid
        print('kfold:', c)
        print('\tmse:', mse)
        print('\tr2:', r2)
        c = c + 1
        mses.append(mse)
        r2s.append(r2)
    mses_mean = sum(mses) / n
    r2s_mean = sum(r2s) / n
    return [mses_mean, r2s_mean]

######################################### Tipos de escalamiento ##################################################################

scale_types = ['none', 'std', 'robust']
type_resgressions = [1, 2, 3]
# learning_rates = ['constant', 'optimal', 'invscaling', 'adaptive']
learning_rates = ['constant']
eta0s = [0.0001, 0.00001, 0.000001]
iterationss = [200000, 300000]

```

Definimos una variable para obtener los mejores parámetros, los tipos de escalamiento y el escalamiento

Código Fuente

```
43  ############################################## Escalamiento ##################################################################
44
45  for scale_type in scale_types:
46      for type_resgression in type_resgressions:
47          for learning_rate in learning_rates:
48              for eta0 in eta0s:
49                  for iterations in iterationss:
50                      print(scale_type)
51                      print('\t', type_resgression)
52                      print('\t\t', learning_rate)
53                      print('\t\t\t', eta0)
54                      print('\t\t\t\t', iterations)
55                      mses_mean, r2s_mean = get_best_params(validation_sets, eta0, learning_rate, iterations, type_resgression
56                      scale_type_list.append(scale_type)
57                      regression_type_list.append(type_resgression)
58                      learning_rate_list.append(learning_rate)
59                      eta0_list.append(eta0)
60                      iterations_list.append(iterations)
61                      mse_list.append(mses_mean)
62                      r2_list.append(r2s_mean)
63                      print("")
64                      print("")
```

Código Fuente

In [9]:

```
# Tabulación

data = {'Tipos de escalamiento': scale_type_list,
        'Tipo de regresión': regression_type_list,
        'Learning': learning_rate_list,
        'eta0': eta0_list,
        'Iteraciones': iterations_list,
        'mse':mse_list,
        'r2':r2_list}
dataframe1 = pd.DataFrame(data)
dataframe1
```

Ahora definimos nuestra tabla para poder obtener los datos con un orden

Código Fuente

```
In [10]:
1  # Tabla ordenada
2  dataframe1.to_csv('resultados.csv')
3  tabla_ordenar = dataframe1.sort_values('mse')
4  tabla_ordenar
```

Out[10]:

Ahora bien, dentro de esta tabla, no vemos un orden lógico, ahora actuamos a ordenarlo por los valores de mse

Código Fuente

In [11]:
```
mse, r2 = polynomial_regression(X_train, X_test, y_train, y_test, degree = 3, eta0 = 0.00001, learning_rate = 'constant', it
```

In [12]:
```
print('mse con X_train y X_test:', mse)
print('r2 con X_train y X_test:', r2)
```

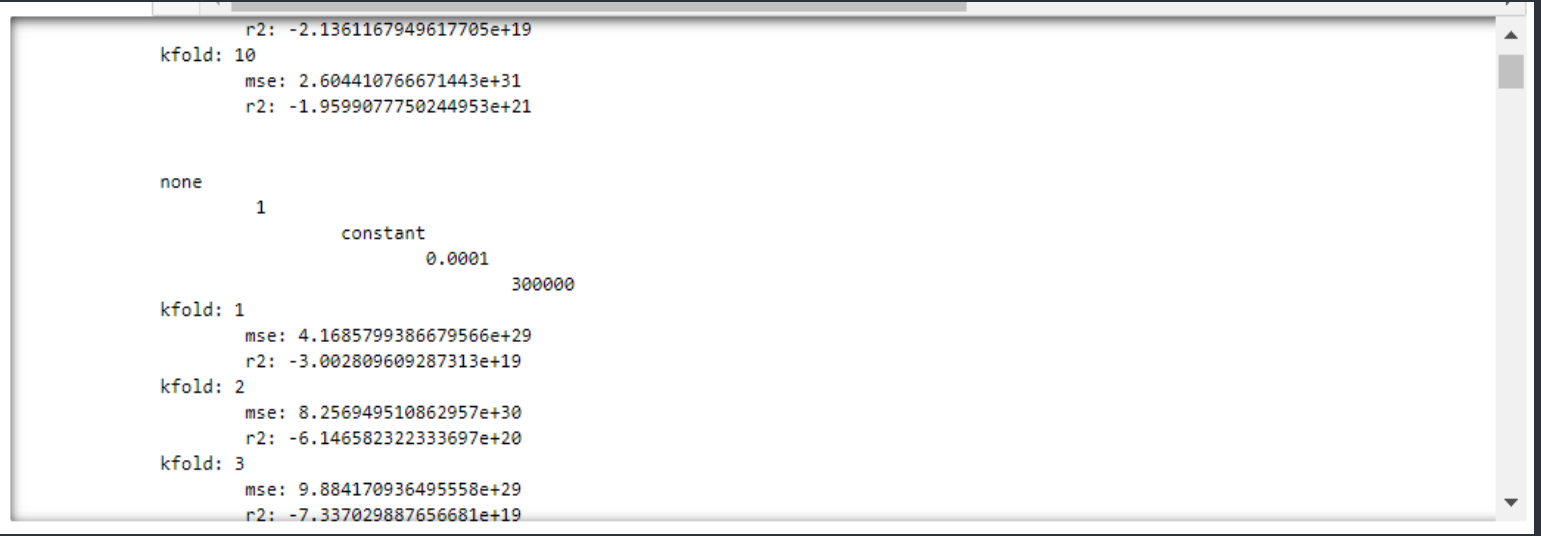
02

{
[Resultados-Tabulación]
}

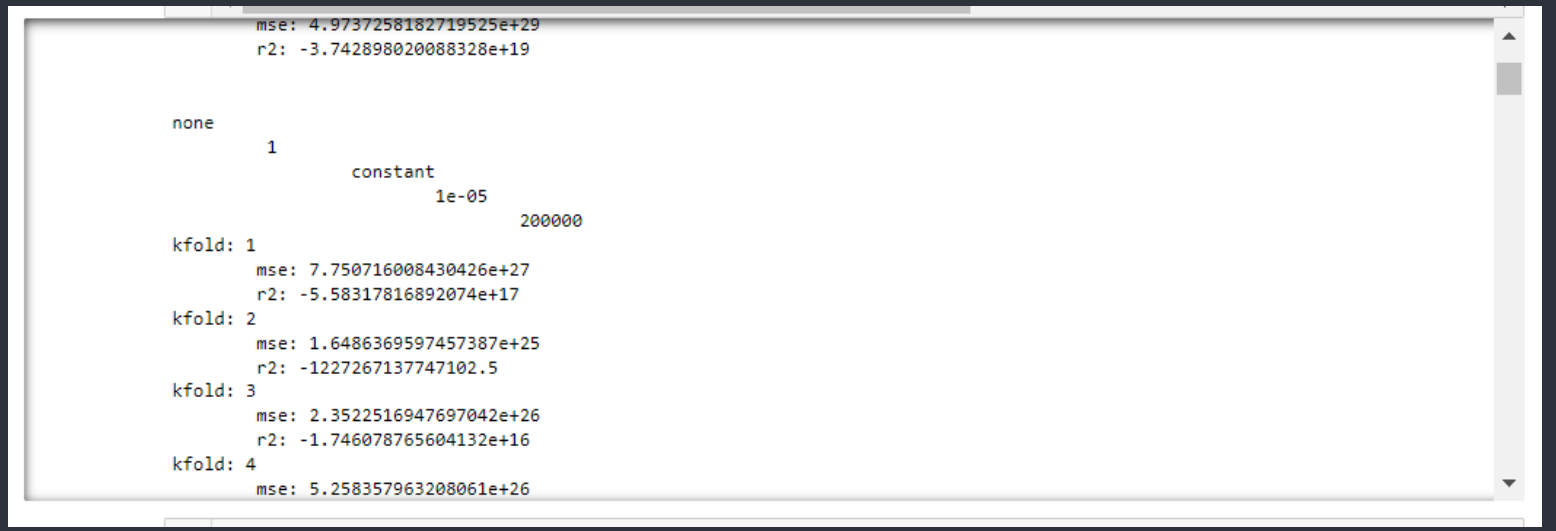
RESULTADOS - PARAMETROS

```
none
        1
                constant
                        0.0001
                                200000
kfold: 1
        mse: 7.192700199140325e+29
        r2: -5.18121508822569e+19
kfold: 2
        mse: 5.56001709344023e+28
        r2: -4.138950193827837e+18
kfold: 3
        mse: 1.6493170640592684e+30
        r2: -1.2242896921727535e+20
kfold: 4
        mse: 1.0782327142660636e+31
        r2: -7.971556722470891e+20
kfold: 5
        mse: 1.309845956394241e+29
```

RESULTADOS - PARAMETROS



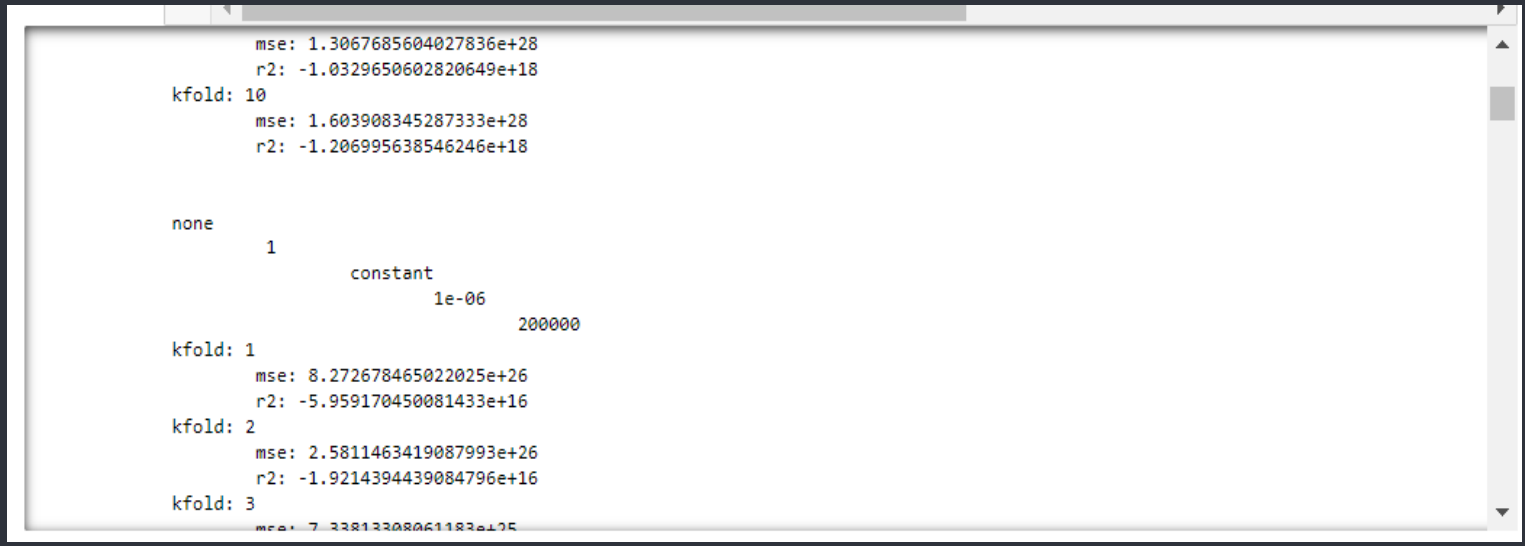
RESULTADOS - PARAMETROS



RESULTADOS - PARAMETROS

```
none
        1
                constant
                        1e-05
                                300000
kfold: 1
        mse: 9.613248908153987e+26
        r2: -6.924841702112051e+16
kfold: 2
        mse: 1.5117898080636857e+27
        r2: -1.125396309751364e+17
kfold: 3
        mse: 4.6675715271335034e+26
        r2: -3.4647429731214104e+16
kfold: 4
        mse: 5.6349983911565445e+26
        r2: -4.1660495653494216e+16
kfold: 5
```

RESULTADOS - PARAMETROS



```
        mse: 1.3067685604027836e+28
        r2: -1.0329650602820649e+18
kfold: 10
        mse: 1.603908345287333e+28
        r2: -1.206995638546246e+18


none
         1
                 constant
                         1e-06
                                 200000
kfold: 1
        mse: 8.272678465022025e+26
        r2: -5.959170450081433e+16
kfold: 2
        mse: 2.5811463419087993e+26
        r2: -1.9214394439084796e+16
kfold: 3
        mse: 7.3381330806118 3e+25
```

RESULTADOS - PARAMETROS

```
kfold: 10
        mse: 6.500778935856916e+24
        r2: -489205748307702.56



none
         1
                 constant
                         1e-06
                                 300000
kfold: 1
        mse: 6.292768366591612e+24
        r2: -453295501063572.56
kfold: 2
        mse: 4.149308682473071e+26
        r2: -3.088800211753892e+16
kfold: 3
        mse: 1.846659452049196e+24
        r2: -137077714246940.9
kfold: 4
```

RESULTADOS - PARAMETROS

```
        r2: -121142192019051.22
kfold: 8
        mse: 5.613946991027348e+24
        r2: -418664666909265.4
kfold: 9
        mse: 2.0876333292182605e+26
        r2: -1.6502174548017188e+16
kfold: 10
        mse: 3.0735165197082735e+23
        r2: -23129258259598.26


robust
         3
                 constant
                         1e-05
                                 200000
kfold: 1
        mse: 9.117600006513666e+22
        r2: -6567804220144.507
```

RESULTADOS – TABULACIÓN SIN ORDENAR

Out[9]:

	Tipos de escalamiento	Tipo de regresión	Learning	eta0	Iteraciones	mse	r2
0	none	1	constant	0.000100	200000	4.620386e+30	-3.451906e+20
1	none	1	constant	0.000100	300000	1.565061e+30	-1.175406e+20
2	none	1	constant	0.000010	200000	6.243362e+27	-4.611774e+17
3	none	1	constant	0.000010	300000	8.296358e+27	-6.384893e+17
4	none	1	constant	0.000001	200000	1.264673e+26	-9.216813e+15
5	none	1	constant	0.000001	300000	1.761400e+26	-1.310978e+16
6	none	2	constant	0.000100	200000	1.705548e+47	-1.262719e+37
7	none	2	constant	0.000100	300000	2.775312e+47	-2.073150e+37
8	none	2	constant	0.000010	200000	3.883418e+45	-2.855971e+35
9	none	2	constant	0.000010	300000	1.320227e+46	-9.511375e+35
10	none	2	constant	0.000001	200000	1.524542e+43	-1.151256e+33
11	none	2	constant	0.000001	300000	1.276942e+43	-9.510445e+32
12	none	3	constant	0.000100	200000	1.122011e+66	-8.250235e+55
13	none	3	constant	0.000100	300000	1.321542e+66	-1.013416e+56
14	none	3	constant	0.000010	200000	2.516924e+63	-1.909706e+53
15	none	3	constant	0.000010	300000	3.556790e+64	-2.799108e+54
16	none	3	constant	0.000001	200000	4.245958e+61	-3.316343e+51
17	none	3	constant	0.000001	300000	2.289084e+61	-1.708496e+51
18	std	1	constant	0.000100	200000	4.823021e+09	6.393440e-01
19	std	1	constant	0.000100	300000	4.824937e+09	6.391974e-01
20	std	1	constant	0.000010	200000	4.822685e+09	6.393663e-01
21	std	1	constant	0.000010	300000	4.822069e+09	6.394121e-01
22	std	1	constant	0.000001	200000	4.821078e+09	6.394871e-01
23	std	1	constant	0.000001	300000	4.820945e+09	6.394966e-01
24	std	2	constant	0.000100	200000	4.430263e+09	6.687677e-01
25	std	2	constant	0.000100	300000	4.411492e+09	6.701361e-01
26	std	2	constant	0.000010	200000	4.433751e+09	6.684951e-01
27	std	2	constant	0.000010	300000	4.427348e+09	6.689693e-01
28	std	2	constant	0.000001	200000	4.413337e+09	6.700165e-01
29	std	2	constant	0.000001	300000	4.412737e+09	6.700638e-01
30	std	3	constant	0.000100	200000	2.256206e+18	-1.664487e+08
31	std	3	constant	0.000100	300000	3.878283e+19	-2.861396e+09
32	std	3	constant	0.000010	200000	4.264831e+09	6.809793e-01
33	std	3	constant	0.000010	300000	4.263254e+09	6.811216e-01
34	std	3	constant	0.000001	200000	4.254667e+09	6.817725e-01
35	std	3	constant	0.000001	300000	4.252322e+09	6.819538e-01
36	robust	1	constant	0.000100	200000	4.900985e+09	6.334022e-01
37	robust	1	constant	0.000100	300000	4.892553e+09	6.340690e-01
38	robust	1	constant	0.000010	200000	4.900094e+09	6.334710e-01
39	robust	1	constant	0.000010	300000	4.898692e+09	6.335806e-01
40	robust	1	constant	0.000001	200000	4.890141e+09	6.342141e-01
41	robust	1	constant	0.000001	300000	4.890130e+09	6.342147e-01
42	robust	2	constant	0.000100	200000	2.300894e+20	-1.712873e+10
43	robust	2	constant	0.000100	300000	2.449452e+19	-1.884110e+09
44	robust	2	constant	0.000010	200000	5.363482e+09	5.980839e-01
45	robust	2	constant	0.000010	300000	5.221100e+09	6.089986e-01
46	robust	2	constant	0.000001	200000	5.062148e+09	6.202077e-01
47	robust	2	constant	0.000001	300000	5.095500e+09	6.176499e-01
48	robust	3	constant	0.000100	200000	8.206808e+25	-6.286588e+15
49	robust	3	constant	0.000100	300000	6.506099e+26	-4.820852e+16
50	robust	3	constant	0.000010	200000	1.323675e+25	-9.869357e+14
51	robust	3	constant	0.000010	300000	8.708027e+24	-6.459236e+14
52	robust	3	constant	0.000001	200000	4.357090e+22	-3.229512e+12
53	robust	3	constant	0.000001	300000	6.341891e+22	-4.858064e+12

RESULTADOS – TABULACIÓN ORDENADA

Out[10]:

	Tipos de escalamiento	Tipo de regresión	Learning	eta0	Iteraciones	mse	r2
35	std	3	constant	0.000001	300000	4.252322e+09	6.819538e-01
34	std	3	constant	0.000001	200000	4.254667e+09	6.817725e-01
33	std	3	constant	0.000010	300000	4.263254e+09	6.811216e-01
32	std	3	constant	0.000010	200000	4.264831e+09	6.809793e-01
25	std	2	constant	0.000100	300000	4.411492e+09	6.701361e-01
29	std	2	constant	0.000001	300000	4.412737e+09	6.700638e-01
28	std	2	constant	0.000001	200000	4.413337e+09	6.700165e-01
27	std	2	constant	0.000010	300000	4.427348e+09	6.689693e-01
24	std	2	constant	0.000100	200000	4.430263e+09	6.687677e-01
26	std	2	constant	0.000010	200000	4.433751e+09	6.684951e-01
23	std	1	constant	0.000001	300000	4.820945e+09	6.394966e-01
22	std	1	constant	0.000001	200000	4.821078e+09	6.394871e-01
21	std	1	constant	0.000010	300000	4.822069e+09	6.394121e-01
20	std	1	constant	0.000010	200000	4.822685e+09	6.393663e-01
18	std	1	constant	0.000100	200000	4.823021e+09	6.393440e-01
19	std	1	constant	0.000100	300000	4.824937e+09	6.391974e-01
41	robust	1	constant	0.000001	300000	4.890130e+09	6.342147e-01
40	robust	1	constant	0.000001	200000	4.890141e+09	6.342141e-01
37	robust	1	constant	0.000100	300000	4.892553e+09	6.340690e-01
39	robust	1	constant	0.000010	300000	4.898692e+09	6.335806e-01
38	robust	1	constant	0.000010	200000	4.900094e+09	6.334710e-01
36	robust	1	constant	0.000100	200000	4.900985e+09	6.334022e-01
46	robust	2	constant	0.000001	200000	5.062148e+09	6.202077e-01
47	robust	2	constant	0.000001	300000	5.095500e+09	6.176499e-01
45	robust	2	constant	0.000010	300000	5.221100e+09	6.089986e-01
44	robust	2	constant	0.000010	200000	5.363482e+09	5.980839e-01
30	std	3	constant	0.000100	200000	2.256206e+18	-1.664487e+08
43	robust	2	constant	0.000100	300000	2.449452e+19	-1.884110e+09
31	std	3	constant	0.000100	300000	3.878283e+19	-2.861396e+09
42	robust	2	constant	0.000100	200000	2.300894e+20	-1.712873e+10
52	robust	3	constant	0.000001	200000	4.357090e+22	-3.229512e+12
53	robust	3	constant	0.000001	300000	6.341891e+22	-4.858064e+12
51	robust	3	constant	0.000010	300000	8.708027e+24	-6.459236e+14
50	robust	3	constant	0.000010	200000	1.323675e+25	-9.869357e+14
48	robust	3	constant	0.000100	200000	8.206808e+25	-6.286588e+15
4	none	1	constant	0.000001	200000	1.264673e+26	-9.216813e+15
5	none	1	constant	0.000001	300000	1.761400e+26	-1.310978e+16
49	robust	3	constant	0.000100	300000	6.506099e+26	-4.820852e+16
2	none	1	constant	0.000010	200000	6.243362e+27	-4.611774e+17
3	none	1	constant	0.000010	300000	8.296358e+27	-6.384893e+17
1	none	1	constant	0.000100	300000	1.565061e+30	-1.175406e+20
0	none	1	constant	0.000100	200000	4.620386e+30	-3.451906e+20
11	none	2	constant	0.000001	300000	1.276942e+43	-9.510445e+32
10	none	2	constant	0.000001	200000	1.524542e+43	-1.151256e+33
8	none	2	constant	0.000010	200000	3.883418e+45	-2.855971e+35
9	none	2	constant	0.000010	300000	1.320227e+46	-9.511375e+35
6	none	2	constant	0.000100	200000	1.705548e+47	-1.262719e+37
7	none	2	constant	0.000100	300000	2.775312e+47	-2.073150e+37
17	none	3	constant	0.000001	300000	2.289084e+61	-1.708496e+51
16	none	3	constant	0.000001	200000	4.245958e+61	-3.316343e+51
14	none	3	constant	0.000010	200000	2.516924e+63	-1.909706e+53
15	none	3	constant	0.000010	300000	3.556790e+64	-2.799108e+54
12	none	3	constant	0.000100	200000	1.122011e+66	-8.250235e+55
13	none	3	constant	0.000100	300000	1.321542e+66	-1.013416e+56

03

{

[Resultados de MSE y R^2]

}

RESULTADOS

```
In [11]: 1 mse, r2 = polynomial_regression(X_train, X_test, y_train, y_test, degree = 3, eta0 = 0.00001, learning_rate = 'constant', it

In [12]: 1 print('mse con X_train y X_test:', mse)
         2 print('r2 con X_train y X_test:', r2)

mse con X_train y X_test: 4281796831.20584
r2 con X_train y X_test: 0.6716299909141945
```

04

[Conclusiones Generales]

Conclusiones

Para la realización de manera eficaz de esta práctica se tuvo que investigar un poco más a fondo con los programas que nos brindó la profesora, la regresión polinomial.
Las principales dificultades al realizar esta práctica se presentaron al tratar de realizar los procesos de gradiente estocásticos.