Esudiante : Elio Erick Ramos Mosquea

Matricula : 2019-8918

Materia: Inteligencia Artificial

Profesor: Jorge Ramon Taveras

Regresion Lineal

Link hacia Google colab (ITLA Email) https://colab.research.google.com/drive/17-jvdR3zJSJYMkpZTg0Oi_VZaIjMgYzg?usp=sharing

Para este proyecto estaremos empleando el uso del Machine Leaning para predecir la cantidad de usuarios que ingresan al contenido de un blog de animales y en que dias.

Tomaremos una cantidad de valores aleatorios basados en la cantidad de personas que ingresan al blog sobre animales.

MAS ADELANTE LOS VALORES:

```
 Un arreglo con los dias (de la semana) previstos para la prueba del programa.
```

```
x = [1,2,3,4,5,6,7]
```

```
 Un arreglo con la cantidad de usuarios que ingresaron para ver el contenido de la pagina.
```

```
y = [200,300,100,500,1021,1029,665]
```

Luego de saber los datos que usaremos externamente entremos ne contexto.

Usameros Python para poder mostrar y ejecutar un programa con la regresion lineal. Usaremos librerias que nos ayudaran a manipular los datos de manera efectiva y util como Numpy para ordenar y obtener valores especiales, sklearn con su clase de Regresion Lineal y mabplotlib para mostrar graficamente los valores.

Bien ya sabiendo vamos a importar estas librerias:

```
import numpy as np
import matplotlib.pyplot as pl
% %config inlineBackend.figure_format = 'svg'
```

Creamos las variables necesarias para nuestro grafico

```
x = [1,2,3,4,5,6,7]
y = [200,300,100,500,1021,1029,665]
```

```
#Dias de la semana
x = np.array(x)
#Usuarios que visitaron la pagina
y = np.array(y)
```

Tenemos los datos del usuarios ahora usaremos algunos valores fundamentales de los datos para poder realizar efectivamente el valor dado.

Descomponemos cada valor de cada una de las ecuaciones.

$$y = mx + b$$

$$m = \frac{\sum x \sum y - n \sum (xy)}{\left(\sum x\right)^2 - n \sum x^2}$$

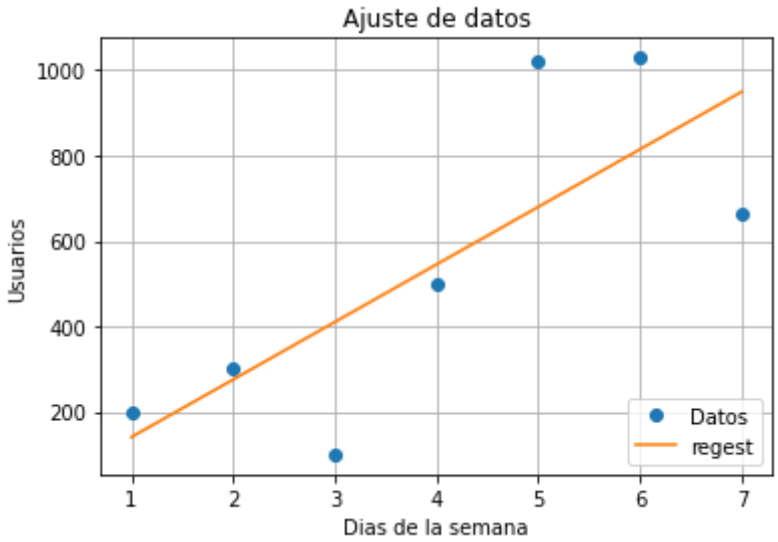
$$b = \bar{y} - m\bar{x}$$

```
#M values
#Tamaño de x's
n = len(x)
#Sumatoria x
sumX = sum(x)
#Sumatoria y
sumY = sum(y)
#Sumatoria x^2
sumX2 = sum(x**2)
#Sumatoria y^2
sumY2 = sum(y**2)
#sumX2, sumX - unnecessary
#Sumatoria x y
sumXY = sum(x*y)


#b values
#Promedio x
promX = sumX / n
#Promedio x
promY = sumY / n


m = (sumX*sumY - n * sumXY) / (sumX**2 - n * sumX2)

b = promY - m * promX
m
```

```
     (134.78571428571428, 5.85714285714289)
```

Bien, con las variables ya asignadas. Podemos hacer un grafico de nuestra regresion lineal.

```
pl.plot(x,y,'o', label='Datos')
pl.plot(x, m * x + b, label='regest')
pl.xlabel('Dias de la semana')
pl.ylabel('Usuarios')
pl.title('Ajuste de datos')
pl.grid()
pl.legend(loc=4)
pl.show()
```



$$R = \frac{\sigma_{xy}}{\sigma_x \sigma_y}$$

$$\sigma_x = \sqrt{\frac{\sum (x^2)}{n} - \bar{x}^2}, \quad \sigma_y = \sqrt{\frac{\sum (y^2)}{n} - \bar{y}^2}$$

$$\sigma_{xy} = \frac{\sum (xy)}{n} - \bar{x} \cdot \bar{y}$$

Ahora vamos a obtener la varianza entre las horas y los usuarios.

```
#Varianza en x
varX = np.sqrt((sumX2 / n) - promX**2)
#Varianza en y
varY = np.sqrt((sumY2 / n) - promY**2)
#Variancia xy
varXY = (sumXY / n) - promX * promY
#R- squared
R = (varXY / (varX * varY))**2
```

```
print('La correlacion de la grafica es % s'%R)
```

```
La correlacion de la grafica es 0.5954140611779567
```

Con el ejemplo dado vamos a verificar que nuestros valores con el uso de la libreria scikit-learn, verficando los valores de arriba.

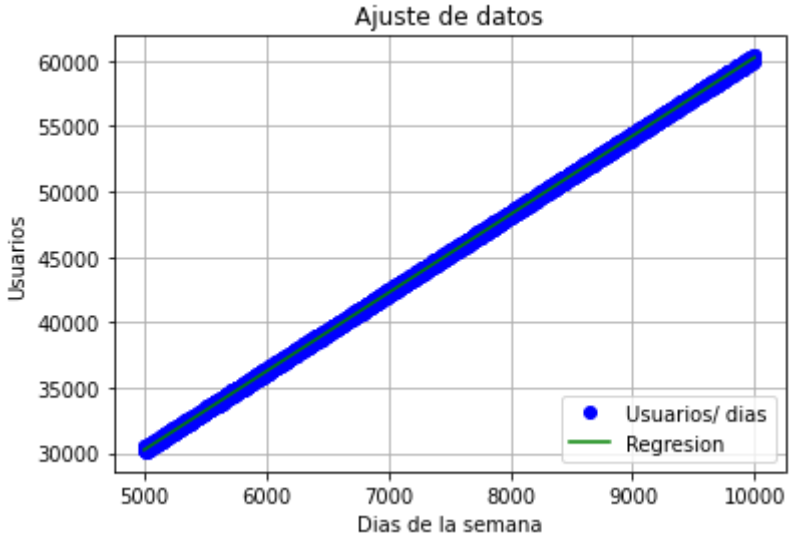
```
x = [1,2,3,4,5,6,7]
y = [200,300,100,500,1021,1029,665]

info = pd.DataFrame({'days': x , 'users': y})

days = info['days'].values.reshape(-1,1)
user = info['users'].values.reshape(-1,1)

days_train = info['days'].values.reshape(-1,1)
days_set = info['days'].values.reshape(-1,1)

user_train = info['users'].values.reshape(-1,1)
user_set = info['users'].values.reshape(-1,1)

#Creo la variable de la regresion lineal
regs = linear_model.LinearRegression()

regs.fit(days_train,user_train)

y_predict = regs.predict(days_set)

# The coefficients
print("Coefficients: \n", regs.coef_)
# The mean squared error
print("Mean squared error: %.2f" % mean_squared_error(user_set, y_predict))
# The coefficient of determination: 1 is perfect prediction
print("Coefficient of determination: %.2f" % r2_score(user_set, y_predict))

#pl.scatter(x_train,y_train,color='r')
pl.plot(days_set,user_set,'o',color= 'b', label='Usuarios/ dias')
pl.plot(days_set,y_predict,color= 'g',label='Regresion')
pl.xlabel('Dias de la semana')
pl.ylabel('Usuarios')
pl.title('Ajuste de datos')
pl.grid()
pl.legend(loc=4)
pl.show()
```

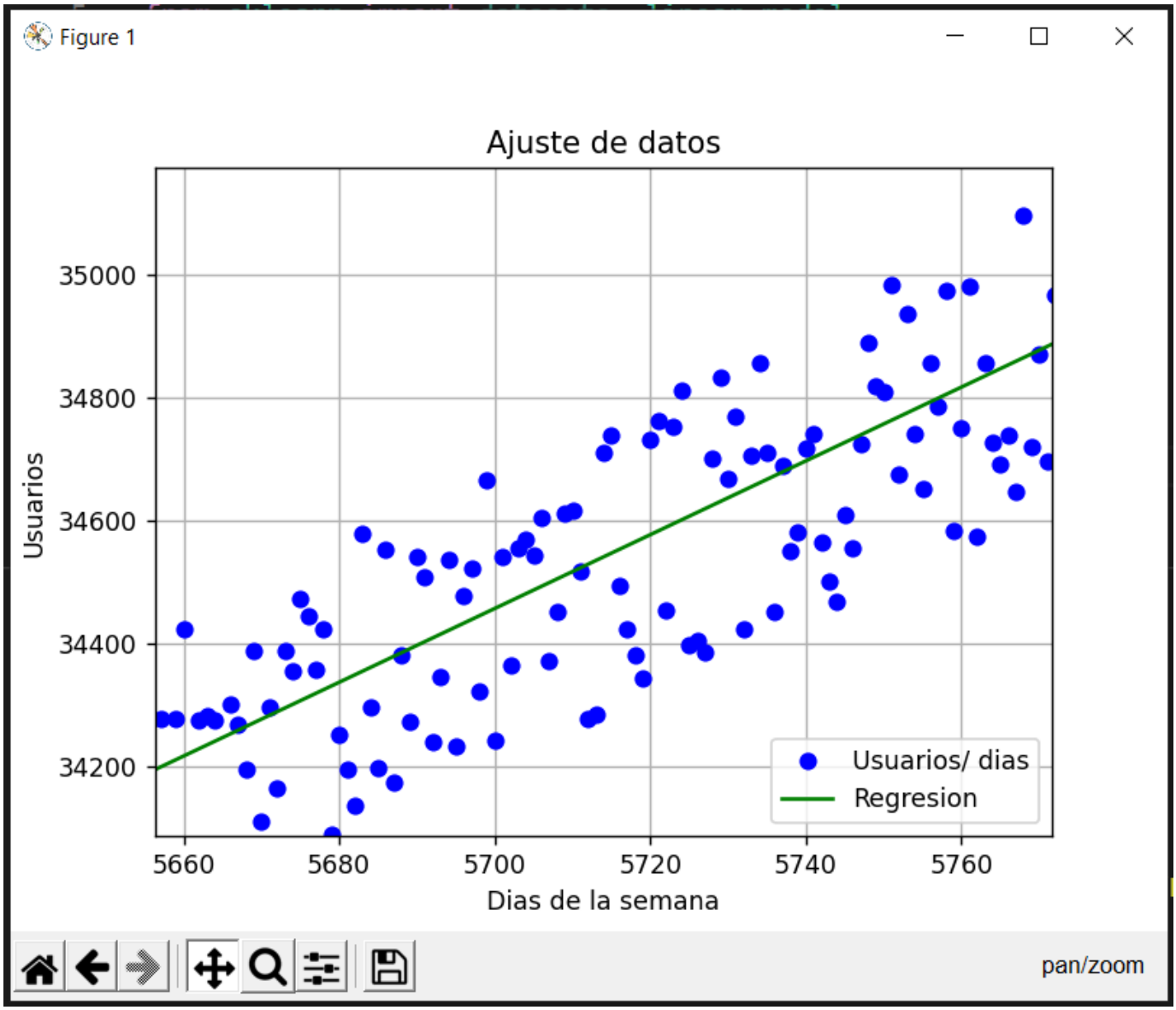
Con esto podemos observar que los valores aproximados concuerdan con los valores de nuestra evaluacion anterior.

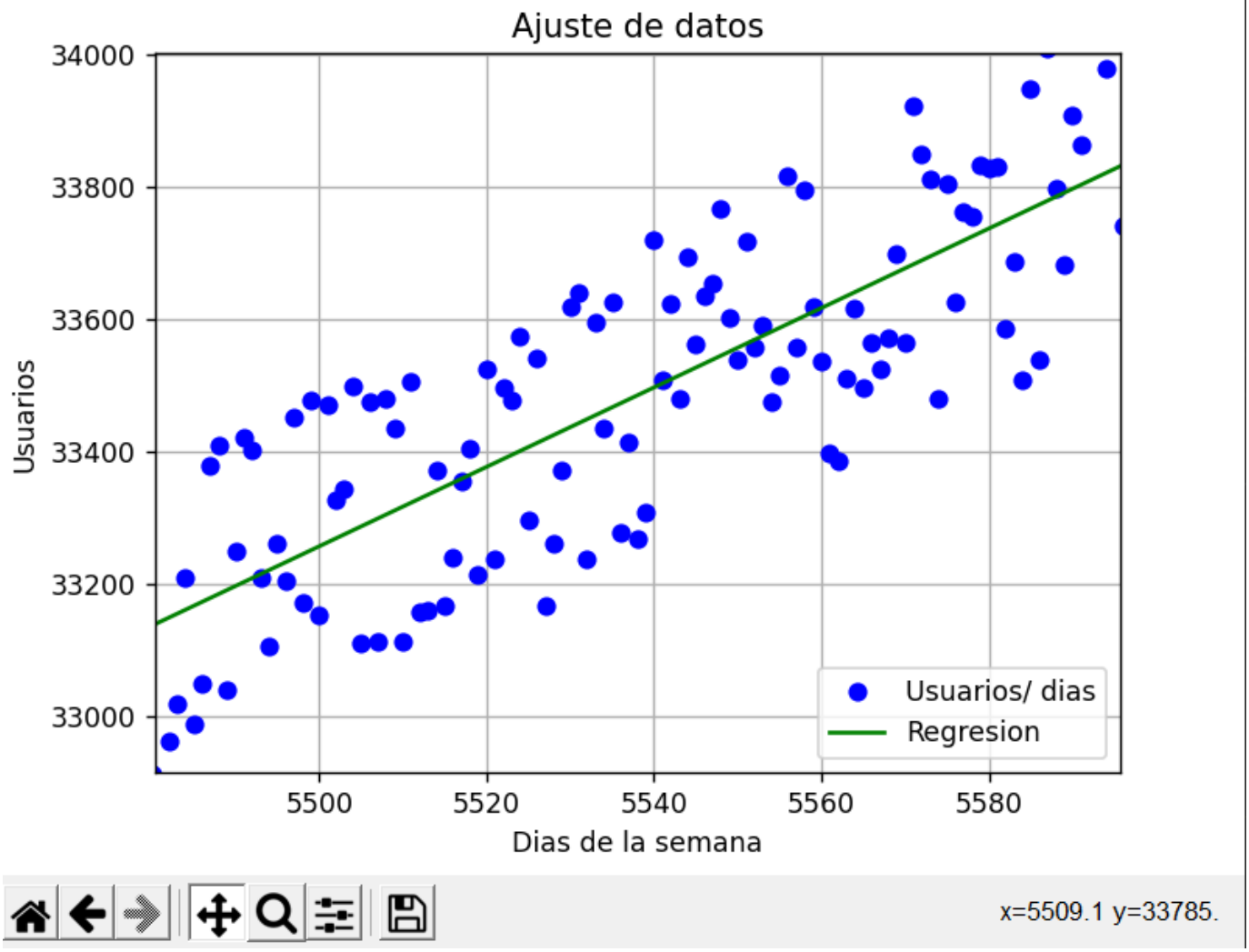
Que tal si esta vez probamos con valores aleatorios.

```
import numpy as np
import random as rdm
import pandas as pd
import matplotlib.pyplot as pl
from sklearn import datasets, linear_model
from sklearn.metrics import mean_squared_error, r2_score

#test
n = 10000
# Cargas los digitos de prueba
x = [r for r in range(n)]
#y = [200,300,100,500,1021,1029,665]
y = [int(rdm.random() * 500) + r * 6  for r in range(n)]

#x = [1,2,3,4,5,6,7]
#y = [200,300,100,500,1021,1029,665]

info = pd.DataFrame({'days': x , 'users': y})

days = info['days'].values.reshape(-1,1)
user = info['users'].values.reshape(-1,1)

days_train = info['days'].values.reshape(-1,1)[:-5000]
days_set = info['days'].values.reshape(-1,1)[-5000:]

user_train = info['users'].values.reshape(-1,1)[:-5000]
user_set = info['users'].values.reshape(-1,1)[-5000:]

#Creo la variable de la regresion lineal
regs = linear_model.LinearRegression()

regs.fit(days_train,user_train)

y_predict = regs.predict(days_set)

# The coefficients
print("Coefficients: \n", regs.coef_)
# The mean squared error
print("Mean squared error: %.2f" % mean_squared_error(user_set, y_predict))
# The coefficient of determination: 1 is perfect prediction
print("Coefficient of determination: %.2f" % r2_score(user_set, y_predict))

#pl.scatter(x_train,y_train,color='r')
pl.plot(days_set,user_set,'o',color= 'b', label='Usuarios/ dias')
pl.plot(days_set,y_predict,color= 'g',label='Regresion')
pl.xlabel('Dias de la semana')
pl.ylabel('Usuarios')
pl.title('Ajuste de datos')
pl.grid()
pl.legend(loc=4)
pl.show()
```

```
Coefficients:
 [[6.00167203]]
Mean squared error: 21155.94
Coefficient of determination: 1.00
```







✓ 0s completed at 11:58 AM