Alumno:	García González Aarón Antonio
Grupo:	3CV9
Unidad de Aprendizaje:	Procesamiento de lenguaje natural
Profesora:	Kolesnikova Olga
Tarea #4:	Analizar y mejorar el programa de regresión lineal
Fecha:	Martes 28 de abril de 2020

Dividir el conjunto de datos en 3 subconjuntos (entrenamiento, validación, pruebas)

```
# Funcion que separa el conjunto de datos original en 3 subconjuntos
# recibe como parametros, matriz de caracteristicas original X,
# al vector de valor objetivo Y,
# el porcentaje de datos que se requiere para entrenamiento
# y el procentaje de datos que serequiere para pruebas
# internamente calcula el porcentaje restante para validacion
def separateSet(X, Y, train_percentage, test_percentage):

    data_zipped = list(zip(X, Y))
    random.shuffle(data_zipped) # revuelve la data
    X, Y = zip(*data_zipped) # descomprime el iterable dado

    total_test = math.ceil(len(Y) * test_percentage)
    total_train = math.ceil(len(Y) * train_percentage)
    total_validate = len(Y) - total_test - total_train

    X_train = X[:total_train]
    X_validate = X[total_train:total_train+total_validate]
    X_test = X[total_train+total_validate:]
    Y_train = Y[:total_train]
    Y_validate = Y[total_train:total_train+total_validate]
    Y_test = Y[total_train+total_validate:]

    return np.array(X_train), np.array(Y_train), np.array(X_validate), np.array(Y_validate),
np.array(X_test), np.array(Y_test)
```

Regularización en modelo de regresión lineal

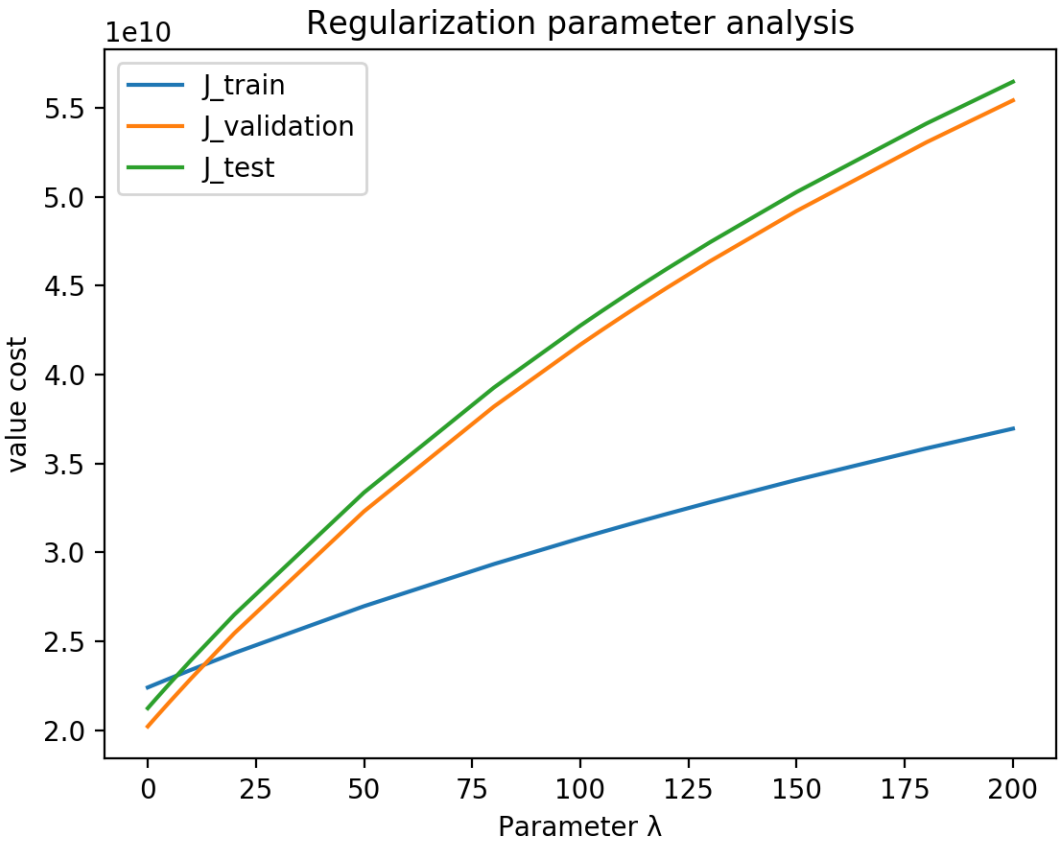
```
# Funcion de hipotesis OK
def h(X):
    return np.dot(X, np.transpose(np.array(θs)))

# Funcion de error OK
def error(X, Y):
    return (h(X)-Y)


# costo
def J(X, Y, λ):
    return (np.sum(error(X, Y)**2) + (λ * np.sum(θs[0:]**2))) / (2*len(Y))


# Funcion J parcial
def J_parcial(X, Y):
    return np.dot(np.transpose(X), error(X, Y)) / len(Y)


# funcion gradiente que actualiza los valores de th
def gradient(X, Y, α, λ):
    global θs

    θs_0_temp = θs[0] - (α * J_parcial(X, Y))[0]
    θs_j_temp = θs[1:]*(1-α*λ/len(Y)) - (α * J_parcial(X, Y))[1:]

    θs[0]  = θs_0_temp
    θs[1:] = θs_j_temp


# Funcion de entrenamiento con descenso gradiente
def train(X,Y,learning_rate,iterations, λ):
    for i in range(0,iterations):
        gradient(X, Y, learning_rate, λ)
```

Pruebas con diferentes valores de parámetro de regularización λ

Se hizo pruebas con diferentes valores del valor λ:

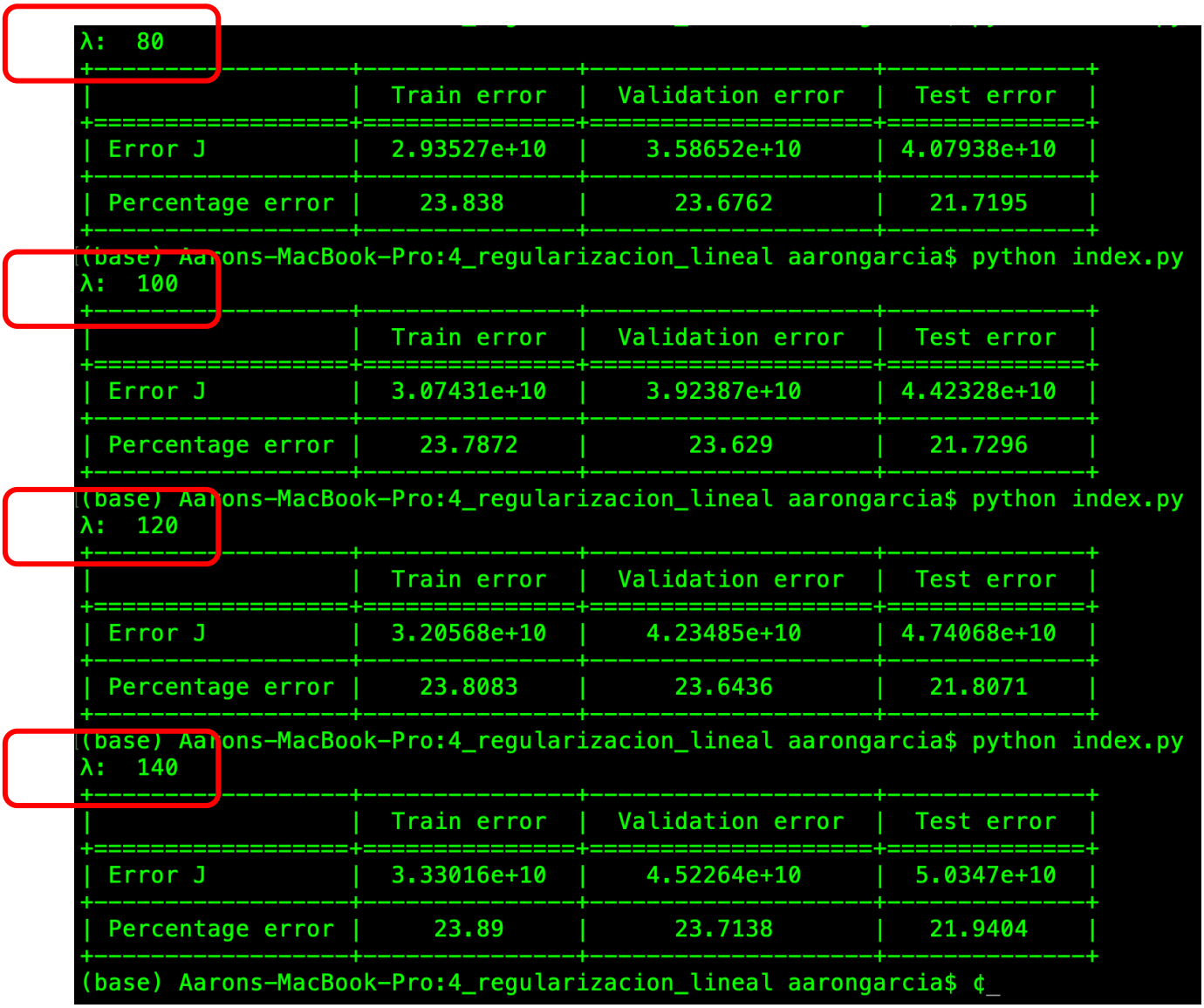
```
λ_values = np.array([0.01,0.1,0.5,0.8,1,2,5,10,20,50,80,100,105,110,112,115,120,130,150,180,200])
```

Con cada uno de estos valores se entreno el modelo de manera separada (es decir, al inicio de cada prueba los valores de los parámetros θ se restauran a 0), y se fueron guardando los valores de error en un arreglo, el valor de error generado fue calculado para cada uno de los 3 subconjuntos en los que se dividió la información (entrenamiento, validación y pruebas).



Como podemos observar en la imagen anterior, tal como lo dice la teoría, el error de validación y prueba será muy similar, por lo que es despreciable cual usar para este ejemplo, usaremos el de validación, si lo ponemos en perspectiva con respecto al error de entrenamiento ahora si observamos diferencias enormes, siguiendo la teoría, puedo decir que cuando el parámetro λ es muy pequeño hay un valor de sesgo muy grande, mismo que se interpreta como **underfit**, y de manera polarizada, cuando λ es mayor a 100 hay un valor de varianza muy grande, que se interpreta como **overfit**, un punto medio donde no hay gran sesgo ni varianza, es en λ ≈ 100.

Ahora nos aproximaremos un poco mas a detalle a λ ≈ 100



En el error de exactitud podemos ver que en λ ≈ 100 al superarse comienza de nuevo a aumentar dicho error.

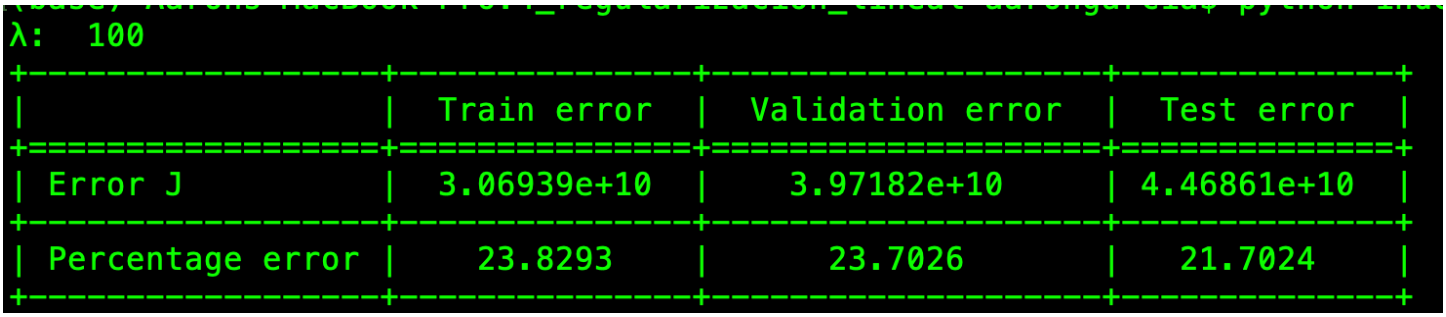
Probar varias sugerencias intentando mejorar los resultados, es decir, disminuir el error

Hay varias técnicas que podemos emplear para minimizar el error, por ejemplo:

- Aumentar conjunto de entrenamiento
- Reducir características
- Encontrar nuevas características
- Aumentar características polinómicas
- Incremento o decremento en el parámetro λ

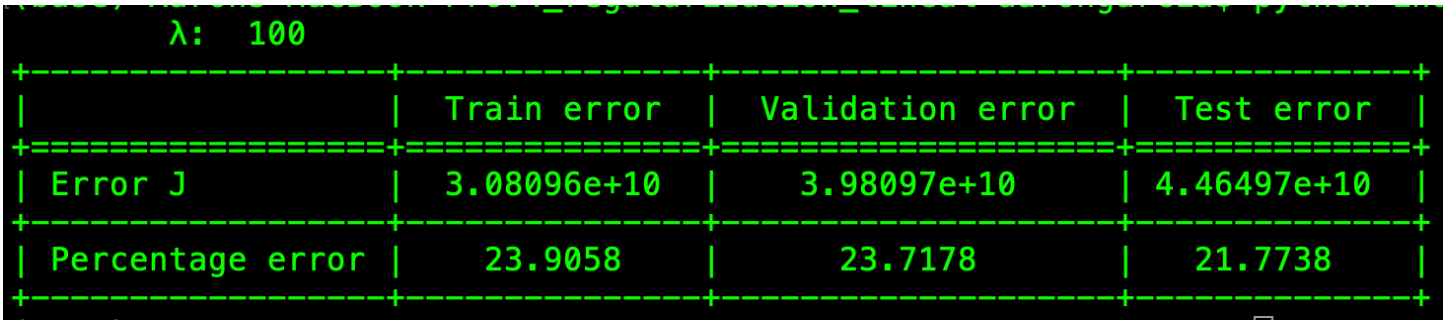
Lo que haremos primeramente será disminuir numero de características, específicamente vamos a quitar:

- Fecha, debido a que el rango de fechas esta en un año, no hay mucha diferencia temporal, por lo que podríamos despreciarlo



Al parecer al quitar esta característica no logramos disminuir el error, al contrario, aumenta drásticamente por lo que esta característica se quedara.

- Ahora probaremos con el código postal, que son números que sin mas contexto no reflejan mucho mas.



Al parecer al quitar esta característica no logramos disminuir el error, al contrario, aumenta ligeramente, por lo que esta característica se quedara.

- Ahora probaremos con el año en que se han renovado las casa, dado que hay muy pocas lo tienen y no muestra el grado de remodelación, pudiese ser solo parcial

```
λ:  100
+------------------+---------------+--------------------+--------------+
|                  |  Train error  |  Validation error  |  Test error  |
+==================+===============+====================+==============+
| Error J          |  3.21346e+10  |    4.02056e+10     |  4.4318e+10  |
+------------------+---------------+--------------------+--------------+
| Percentage error |    25.2119    |      24.9276       |   24.4436    |
+------------------+---------------+--------------------+--------------+
```

Al parecer al quitar esta característica no logramos disminuir el error, al contrario, aumenta medianamente, por lo que esta característica se quedara.

- Ahora voy a probar quitando latitud y longitud, sabemos que están en New York, obviamente hay zonas mas casras que otras, pero los números por si solos de latitud y longitud tal vez no reflejen realmente su ubicación, sino que habría que hacerle feature scaling de manera particular.

```
λ:  100
+------------------+---------------+--------------------+--------------+
|                  |  Train error  |  Validation error  |  Test error  |
+==================+===============+====================+==============+
| Error J          |   3.384e+10   |    4.30595e+10     | 4.86994e+10  |
+------------------+---------------+--------------------+--------------+
| Percentage error |    29.307     |      29.4566       |    26.876    |
+------------------+---------------+--------------------+--------------+
```

Al parecer al quitar esta característica no logramos disminuir el error, al contrario, inesperadamente influyen demasiado estas dos características, ya que los valores se ven muy afectados.

En este caso el numero de características son 19, es decir la característica 19 habría que elevarla a la 19, por lo que no es una opción, tampoco aumentar el numero de datos, porque hay que realizar una gran variedad de datos y eso es tiempo, ya modificamos también el parámetro de regularización, le mejor resultado obtenido fue el que fue obtenido con $\lambda \approx 100$;