

# Tarea2

July 18, 2017

## 1 Proyecto 2: Propedéutico

Alumno: Jorge III Altamirano Astorga

### 1.1 Parte 1: Teoría de Álgebra Lineal y Optimización

1. ¿Por qué una matriz equivale a una transformación lineal entre espacios vectoriales?

Un vector puede manejarse como una matriz con shape  $(n,1)$  o un arreglo uni-dimensional, dado que es una entidad con cantidad y dirección. Computacionalmente, una matriz es un arreglo multidimensional; mientras que un vector es el caso de una matriz unidimensional.

*Necesito estudiar las siguientes preguntas*

2. ¿Cuál es el efecto de transformación lineal de una matriz diagonal y el de una matriz ortogonal?
3. ¿Qué es la descomposición en valores singulares de una matriz?
4. ¿Qué es diagonalizar una matriz y que representan los eigenvectores?
5. ¿Intuitivamente qué son los eigenvectores?
6. ¿Cómo interpretas la descomposición en valores singulares como una composición de tres tipos de transformaciones lineales simples?
7. ¿Qué relación hay entre la descomposición en valores singulares y la diagonalización?
8. ¿Cómo se usa la descomposición en valores singulares para dar una aproximación de rango menor a una matriz?
9. Describe el método de minimización por descenso gradiente
10. Menciona 4 ejemplo de problemas de optimización (dos con restricciones y dos sin restricciones) que te parecan interesantes como Científico de Datos

### 1.2 Parte 2: Aplicaciones en Python

```
In [1]: import numpy as np
        from PIL import Image
        import matplotlib.pyplot #import matplotlib.pyplot as plt
```

```
img = Image.open("tarea2.color.png")  
img
```

Out[1]:



```
In [2]: img = img.convert("LA")  
        row, col = img.size  
        data = np.zeros((row*col, 5))  
        matplotlib.pyplot.figure(figsize=(9, 6))  
        matplotlib.pyplot.imshow(img)  
        img
```

Out[2]:



### 1.2.1 1. SVD

```
In [3]: A = np.matrix(img.getdata(band=0), float)
        A
```

```
Out[3]: matrix([[ 171.,  180.,  176., ...,  64.,  65.,  63.]])
```

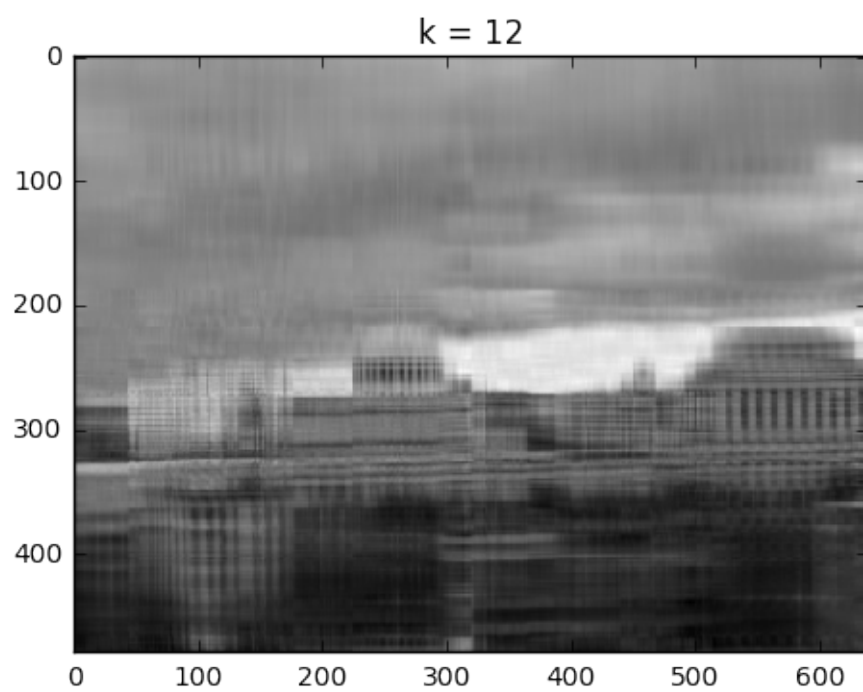
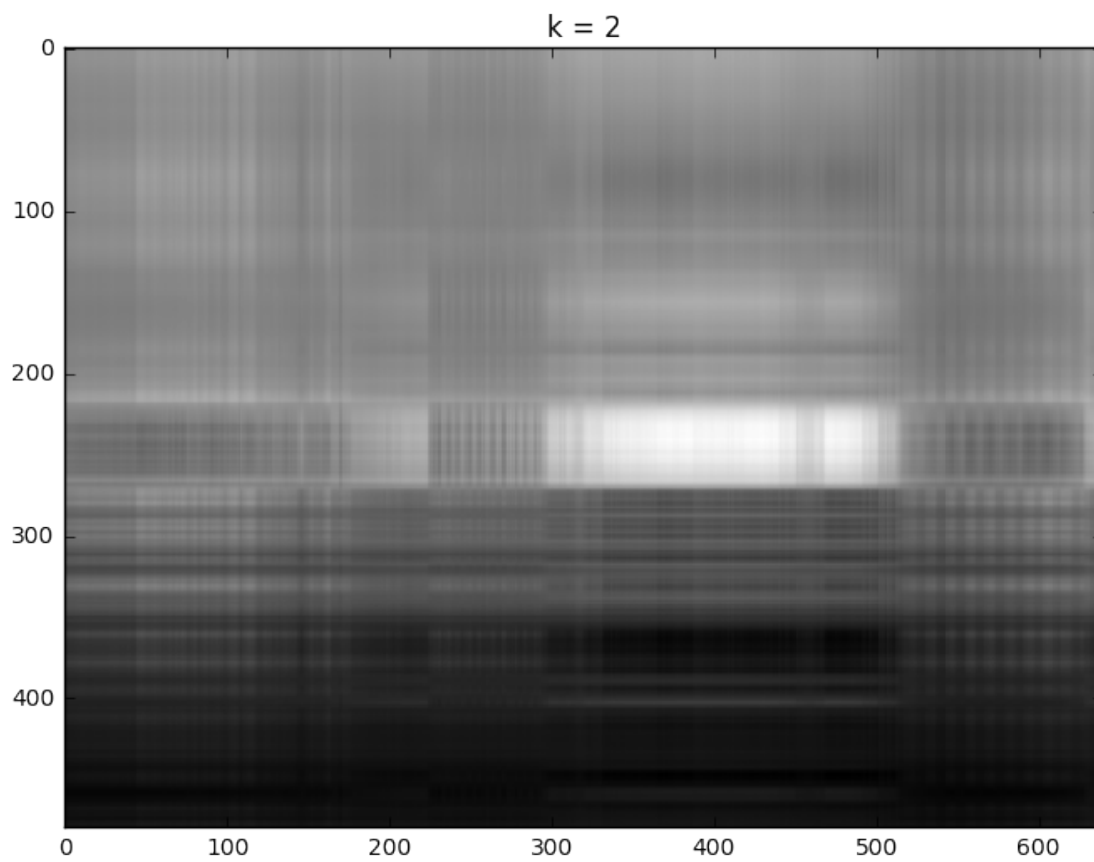
```
In [4]: A.shape = (img.size[1], img.size[0])
        A.size
```

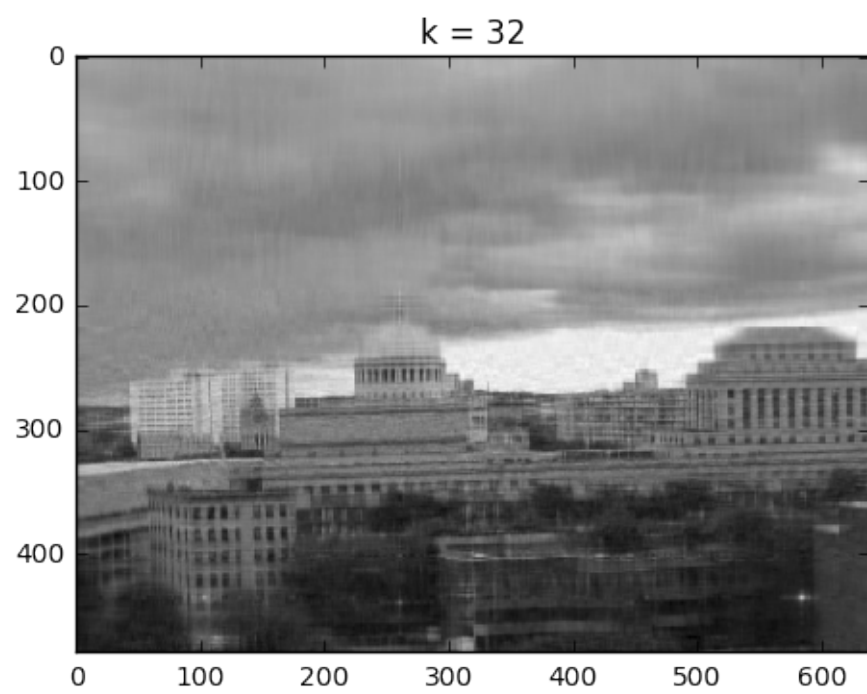
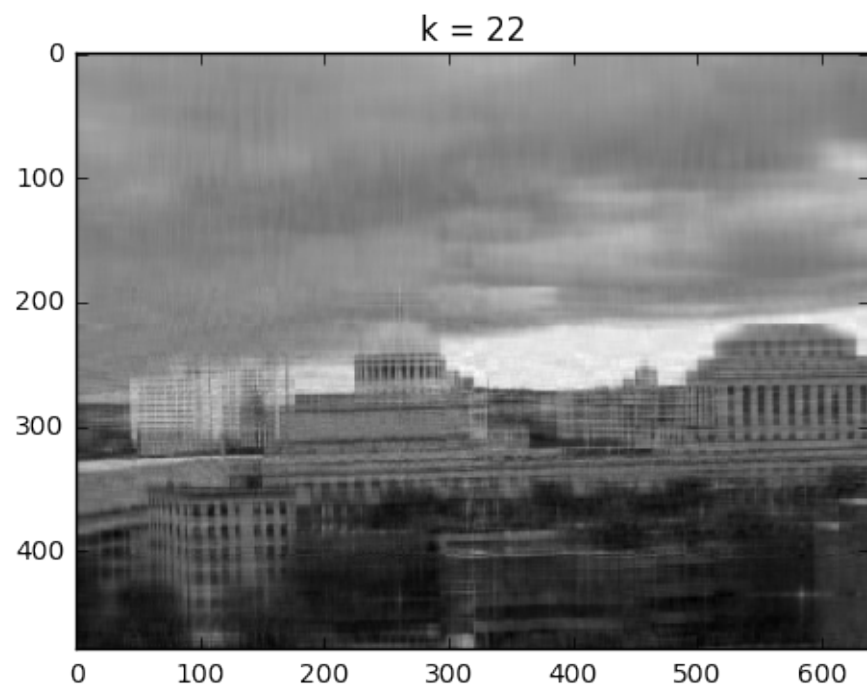
```
Out[4]: 307200
```

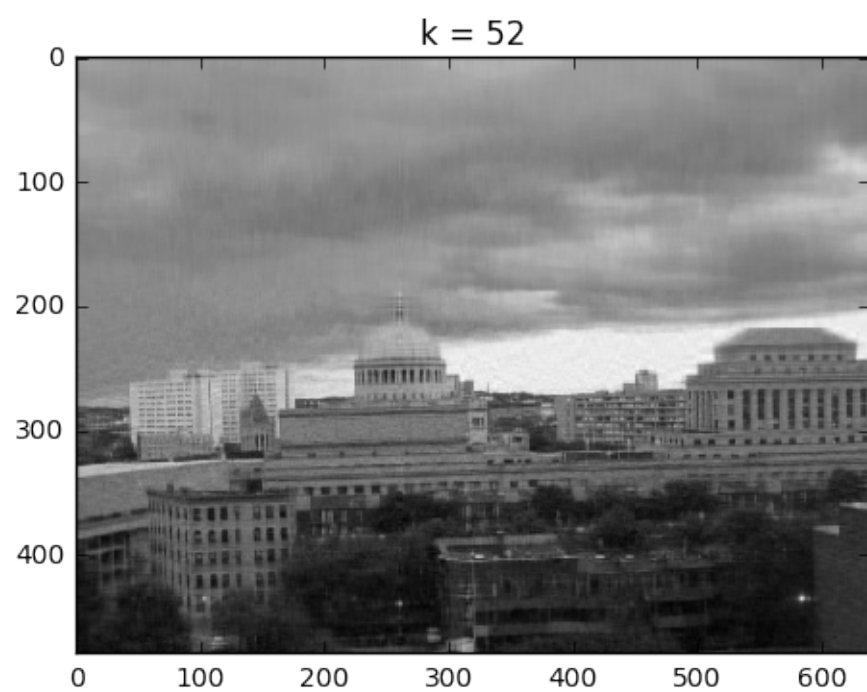
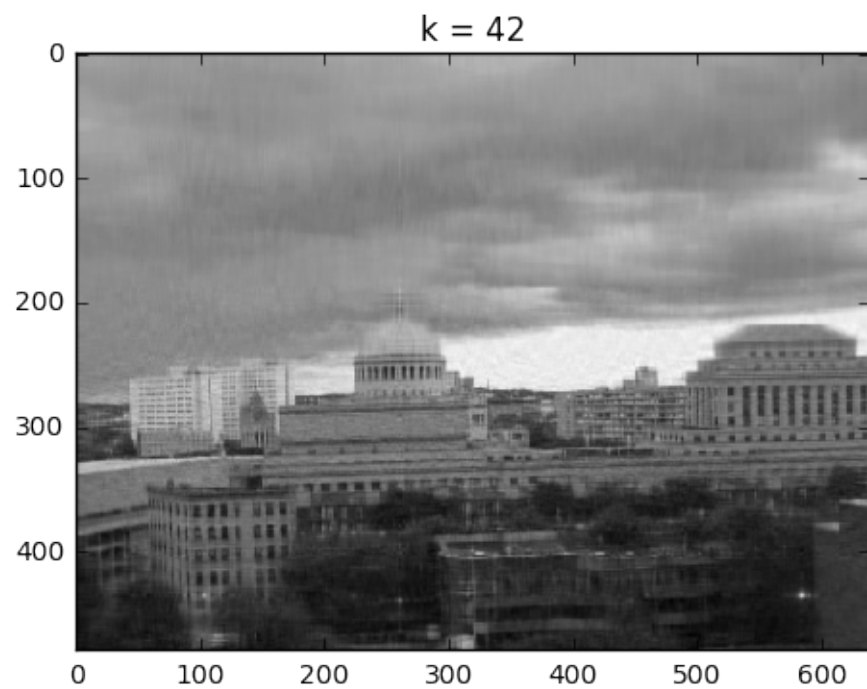
```
In [5]: u, s, v = np.linalg.svd(A, full_matrices=1)
```

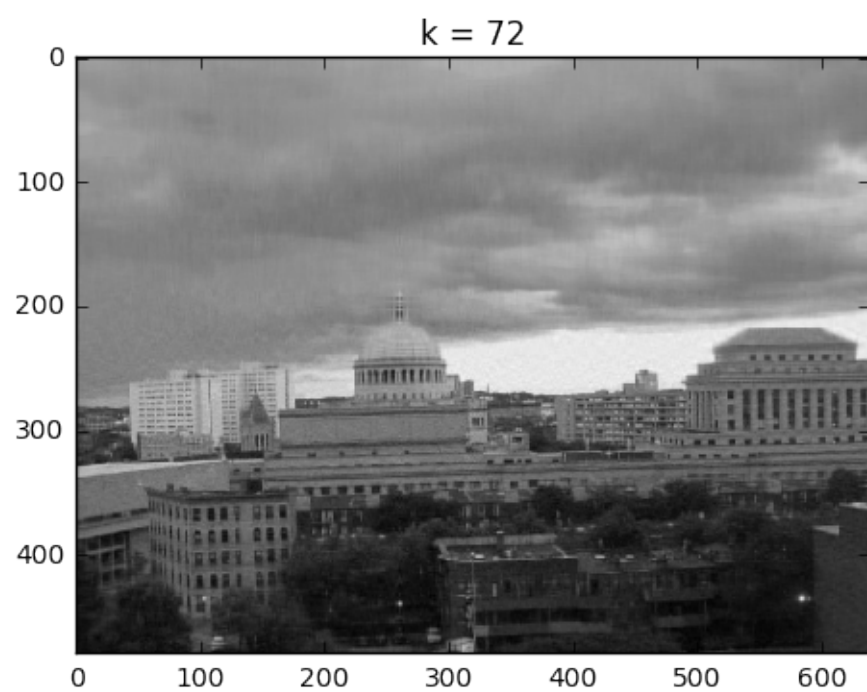
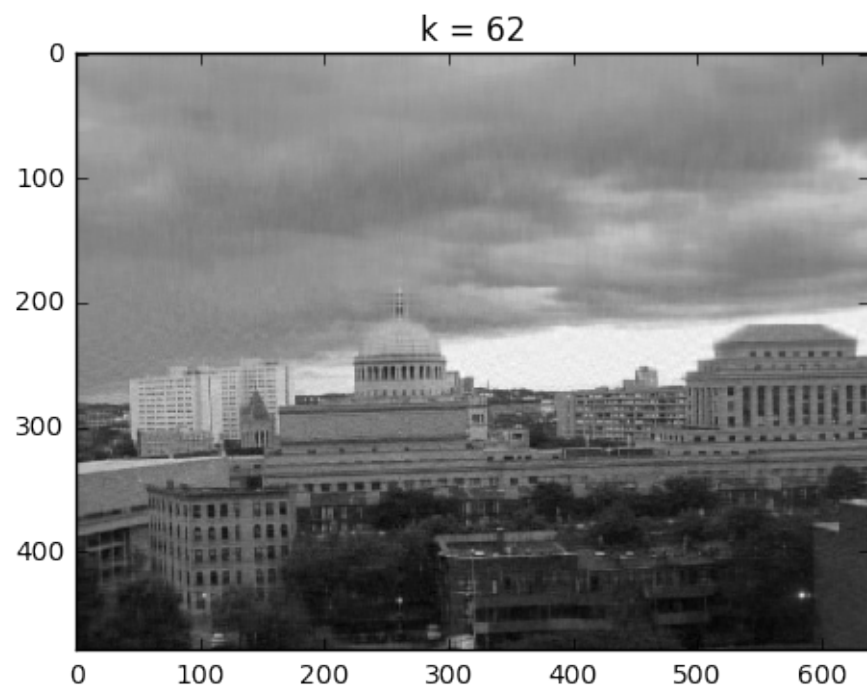
```
In [6]: matplotlib.pyplot.figure(figsize=(9,6))
        matplotlib.pyplot.imshow(img, cmap='gray')
        for k in range(2,100,10):
            B = np.matrix(u[:, :k]) * np.diag(s[:k]) * np.matrix(v[:,k, :])
            matplotlib.pyplot.imshow(B, cmap='gray')
            matplotlib.pyplot.title("k = " + str(k))
            matplotlib.pyplot.show()
```



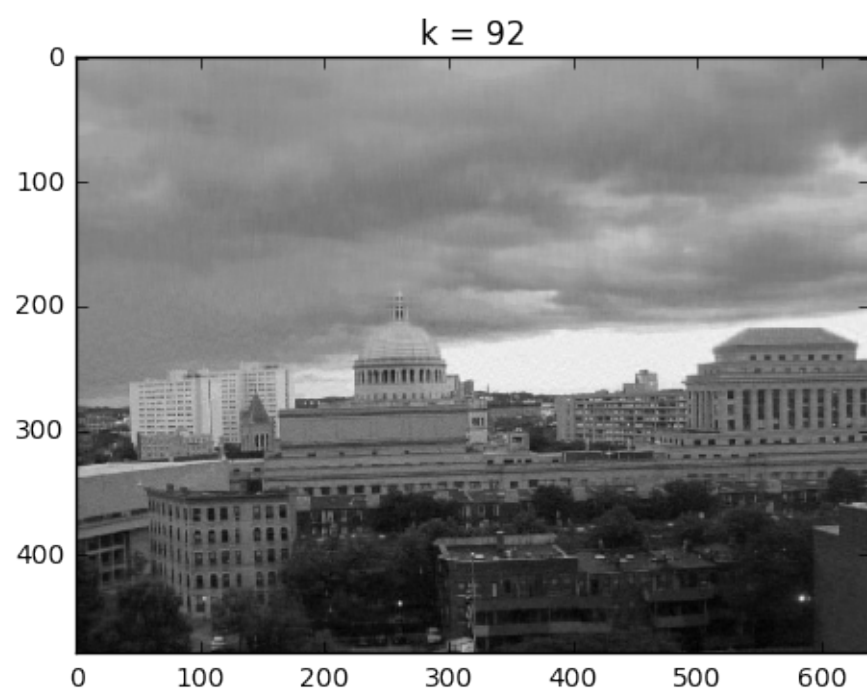
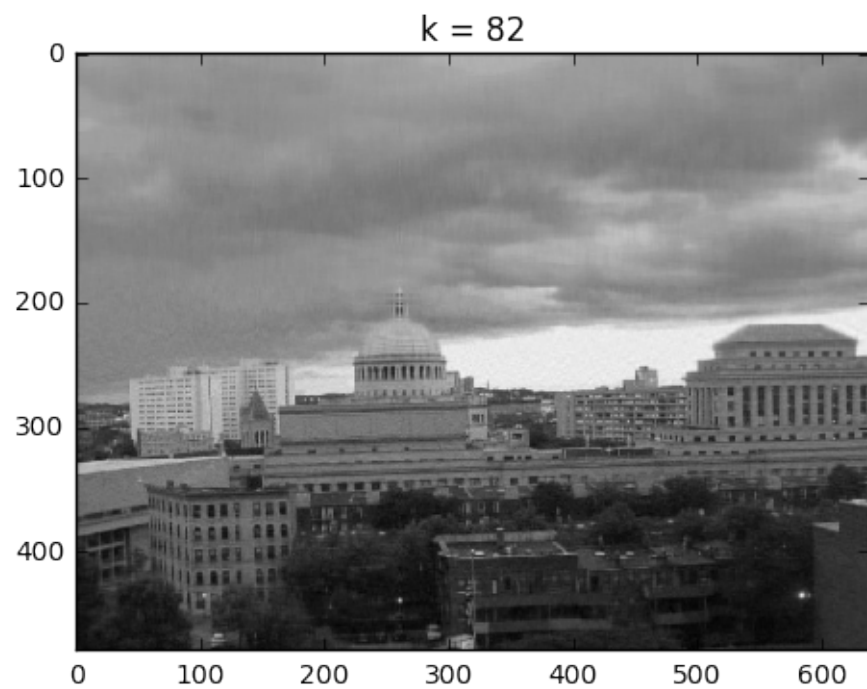












¿Qué tiene que ver con la compresión de imágenes? Como se puede observar, muchos algoritmos utilizan la compresión a través de la operación SVD, pues reduce el número de columnas, renglones en una matriz y va reduciéndose de manera progresiva el número de renglones y en consecuencia del tamaño de archivos, por ende perdiendo calidad.

### 1.2.2 a. Pseudoinversa

```
In [7]: import numpy as np

def pinversa(A):
    u, s, v = np.linalg.svd(A)
    B = np.dot(v.transpose(), np.dot(np.diag(s**-1), u.transpose()))
    B = v.transpose() * (np.diag(s**-1)*u.transpose())
    return np.array(B)

A = np.matrix([[1,2,3],[4,5,6],[7,8,9]])
C = np.matrix([[1,2],[4,5]])
C = pinversa(C)
C

Out[7]: array([[ -1.66666667,  0.66666667],
               [ 1.33333333, -0.33333333]])
```

Se muestra como al aplicar la inversa nuevamente vuelve a su forma original

```
In [8]: np.linalg.pinv(C)

Out[8]: array([[ 1.,  2.],
               [ 4.,  5.]])
```

### 1.2.3 3. Pandas

```
In [15]: import pandas as pd
import os
import numpy as np
import statsmodels.formula.api as sm

studysat = pd.read_csv("study_vs_sat.csv")
studysat = pd.DataFrame(studysat)
studysat

Out[15]:
```

	study_hours	sat_score
0	4	390
1	9	580
2	10	650
3	14	730
4	4	410
5	7	530
6	12	600

7	22	790
8	1	350
9	3	400
10	8	590
11	11	640
12	5	450
13	6	520
14	10	690
15	11	690
16	16	770
17	13	700
18	13	730
19	10	640

```
In [28]: modelo = sm.ols("study_hours ~ sat_score", studysat)
         modelo = modelo.fit()
         modelo.params
```

```
Out[28]: Intercept    -10.941095
         sat_score      0.034415
         dtype: float64
```

```
In [35]: sumhours = sum(studysat["study_hours"])
         sumsat = sum(studysat["sat_score"])
         sumhourssat = sum(studysat["study_hours"] * studysat["sat_score"])
         sumhours2 = sum(studysat["study_hours"]**2)
         sumsat2 = sum(studysat["sat_score"]**2)
         length = len(studysat["study_hours"])
         = (length * sumhourssat - sumhours*sumsat)/(length * sumhours2 - sumhours**2)
```

```
Out[35]: 25.326467777895743
```

```
In [36]: = sumsat / length - (*sumhours) / length
```

```
Out[36]: 353.16487949888523
```