

# Tarea evaluable

## CE\_5075 7.1

Big data aplicado




## APARTADO 4

Crea un cuaderno de Google Colab que, utilizando los servidores de Google, realice todo lo que se solicita en el apartado 3. Debes utilizar PySpark y RDDs. Asegúrate de que el archivo *preus.txt* se lea desde tu unidad de Google Drive.

Una vez más, adjunto algunas capturas del código del cuaderno pero añadido el archivo de Jupyter entre los archivos de la entrega.


```
[1] from pyspark import SparkContext

# Inicializar SparkContext
sc = SparkContext("local", "PreciosAirbnbGoogleColab")
sc
```

 **SparkContext**  
[Spark UI](#)  
Version  
v3.5.5  
Master  
local  
AppName  
PreciosAirbnbGoogleColab








```
[2] from google.colab import drive

# Montar Google Drive
DRIVE = '/content/drive'
drive.mount(DRIVE)
```


 Mounted at /content/drive

```
# Cargar datos desde el archivo montado en Drive
rdd = sc.textFile(f'{DRIVE}/MyDrive/Curso IA & Big Data/Big data aplicado/preus.txt')


# Convertir los datos a tipo float y eliminar encabezados y valores no numéricos
prices_rdd = rdd.filter(lambda x: x.strip().isdigit()).map(lambda x: float(x.strip()))
```


```
[4] # Número de alojamientos
num_accommodations = prices_rdd.count()
print(f'Número de alojamientos: {num_accommodations}')
```



 Número de alojamientos: 1048253

```
[5] # Valor mínimo y máximo de precios
min_price = prices_rdd.min()
max_price = prices_rdd.max()
print(f'Mínimo: {min_price} - Máximo: {max_price}')
```

 Mínimo: 10.0 - Máximo: 23229.0

```
[6] # Media y desviación estándar
mean_price = prices_rdd.mean()
stdev_price = prices_rdd.stdev()
print(f'Media: {mean_price} - Desviación estándar: {stdev_price}')
```



 + Código  + Texto

```
print(f'Media: {mean_price} - Desviación estándar: {stdev_price}')
```

Media: 436.0350125399073 - Desviación estándar: 1188.3904253821997

✓ 0 s [7] # Actualizar los valores de precio  
new\_price\_rdd = prices\_rdd.map(lambda x: x \* 1.02 if x < 1000 else x \* 1.03)

✓ 12 s [8] # Nueva media y desviación estándar  
new\_mean\_price = new\_price\_rdd.mean()  
new\_stdev\_price = new\_price\_rdd.stdev()  
print(f'NUEVA Media: {new\_mean\_price} - NUEVA Desviación estándar: {new\_stdev\_price}')

NUEVA Media: 446.8178788517739 - NUEVA Desviación estándar: 1224.0973352938518

✓ 11 s # Contar alojamientos por grupos  
price\_groups = {  
 'Grupo 1': new\_price\_rdd.filter(lambda x: x <= 150).count(),  
 'Grupo 2': new\_price\_rdd.filter(lambda x: 151 <= x <= 300).count(),  
 'Grupo 3': new\_price\_rdd.filter(lambda x: 301 <= x <= 500).count(),  
 'Grupo 4': new\_price\_rdd.filter(lambda x: x > 500).count()  
}  
  
for group, count in price\_groups.items():  
 print(f'{group}: {count}')

Grupo 1: 396986  
Grupo 2: 300880  
Grupo 3: 145719  
Grupo 4: 199829