Dmytro Mahaliuk
55722

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import scipy.stats as scs
```

1.

```
df = pd.DataFrame({'x':[1, 2, 3, 4, 5], 'y':['a', 'b', 'a', 'b', 'b']})
group = df.groupby('y')
res = group['x'].mean()

print("\n",res)
```

```
 y
a    2.000000
b    3.666667
Name: x, dtype: float64
```

2.

```
x_count = df['x'].value_counts()
y_count = df['y'].value_counts()

print("\n",x_count)
print("\n",y_count)
```

```
 x
1    1
2    1
3    1
4    1
5    1
Name: count, dtype: int64

 y
b    3
a    2
Name: count, dtype: int64
```

3.

```
np_autos = np.loadtxt( fname: 'autos.csv', delimiter=',', skiprows=1, dtype=str)
df = pd.read_csv('autos.csv')

print("\n",np_autos[:5])
print("\n",df.head())
```

```
[['0' '' 'alfa-romero' 'gas' 'std' 'two' 'convertible' 'rwd' 'front'
  '88.6' '168.8' '64.1' '48.8' '2548.0' 'dohc' 'four' '130.0' 'mpfi'
  '3.47' '2.68' '9.0' '111.0' '5000.0' '21.0' '27.0' '13495.0' '3']
 ['1' '' 'alfa-romero' 'gas' 'std' 'two' 'convertible' 'rwd' 'front'
  '88.6' '168.8' '64.1' '48.8' '2548.0' 'dohc' 'four' '130.0' 'mpfi'
  '3.47' '2.68' '9.0' '111.0' '5000.0' '21.0' '27.0' '16500.0' '3']
 ['2' '' 'alfa-romero' 'gas' 'std' 'two' 'hatchback' 'rwd' 'front' '94.5'
  '171.2' '65.5' '52.4' '2823.0' 'ohcv' 'six' '152.0' 'mpfi' '2.68'
  '3.47' '9.0' '154.0' '5000.0' '19.0' '26.0' '16500.0' '1']
 ['3' '164.0' 'audi' 'gas' 'std' 'four' 'sedan' 'fwd' 'front' '99.8'
  '176.6' '66.2' '54.3' '2337.0' 'ohc' 'four' '109.0' 'mpfi' '3.19' '3.4'
  '10.0' '102.0' '5500.0' '24.0' '30.0' '13950.0' '2']
 ['4' '164.0' 'audi' 'gas' 'std' 'four' 'sedan' '4wd' 'front' '99.4'
  '176.6' '66.4' '54.3' '2824.0' 'ohc' 'five' '136.0' 'mpfi' '3.19' '3.4'
  '8.0' '115.0' '5500.0' '18.0' '22.0' '17450.0' '2']]

   Unnamed: 0  normalized-losses         make  ... highway-mpg    price symboling
0           0                NaN  alfa-romero  ...        27.0  13495.0         3
1           1                NaN  alfa-romero  ...        27.0  16500.0         3
2           2                NaN  alfa-romero  ...        26.0  16500.0         1
3           3              164.0         audi  ...        30.0  13950.0         2
4           4              164.0         audi  ...        22.0  17450.0         2

[5 rows x 27 columns]
```

numpy.loadtxt()
Działa głównie na danych numerycznych.
Wymaga, aby wszystkie kolumny miały ten sam typ danych
Nie obsługuje dobrze nagłówków (można pominąć pierwszą linię, ale nie jest to natywne).

pandas.read_csv()
Obsługuje zarówno dane numeryczne, jak i tekstowe.
Może automatycznie rozpoznawać typy danych w kolumnach.
Obsługuje nagłówki, brakujące wartości, różne separatory, kodowania znaków itp.

4.

```
group = df.groupby('make')
group_mean = group[['city-mpg', 'highway-mpg']].mean()
group_mean['average-mpg'] = group_mean.mean(axis=1)

print("\n",group_mean[['average-mpg']])
```

```
                average-mpg
 make
 alfa-romero      23.500000
 audi             21.500000
 bmw              22.375000
 chevrolet        43.666667
 dodge            31.055556
 honda            32.923077
 isuzu            33.500000
 jaguar           16.333333
 mazda            28.823529
 mercedes-benz    19.750000
 mercury          21.500000
 mitsubishi       28.038462
 nissan           29.972222
 peugot           24.545455
 plymouth         31.142857
 porsche          21.700000
 renault          27.000000
 saa              23.833333
 subaru           28.541667
 toyota           30.203125
 volkswagen       31.750000
 volvo            23.500000
```

5.

```
group = df.groupby('make')['fuel-type'].value_counts()

print("\n",group)
```

```
 make           fuel-type
alfa-romero     gas           3
audi            gas           7
bmw             gas           8
chevrolet       gas           3
dodge           gas           9
honda           gas          13
isuzu           gas           4
jaguar          gas           3
mazda           gas          15
                diesel        2
mercedes-benz   diesel        4
                gas           4
mercury         gas           1
mitsubishi      gas          13
nissan          gas          17
                diesel        1
peugot          gas           6
                diesel        5
plymouth        gas           7
porsche         gas           5
renault         gas           2
saa             gas           6
subaru          gas          12
toyota          gas          29
                diesel        3
volkswagen      gas           8
                diesel        4
volvo           gas          10
                diesel        1
Name: count, dtype: int64
```

6-8.

```
x = df['length'].dropna()
y = df['city-mpg'].dropna()

coeff_1 = np.polyfit(x, y,  deg: 1)
coeff_2 = np.polyfit(x, y,  deg: 2)

print("\n", coeff_1)
print("\n", coeff_2)

sorted_x = np.sort(x)
pred_1 = np.polyval(coeff_1, sorted_x)
pred_2 = np.polyval(coeff_2, sorted_x)

print("\n", pred_1[:5])
print("\n", pred_2[:5])
```

```
[-0.35576533 87.14020723]

[ 4.39610791e-03 -1.89441985e+00  2.21104093e+02]

[36.94171944 35.69654079 35.69654079 33.77540802 33.77540802]

[41.32448739 39.089906   39.089906   35.8535431  35.8535431 ]
```

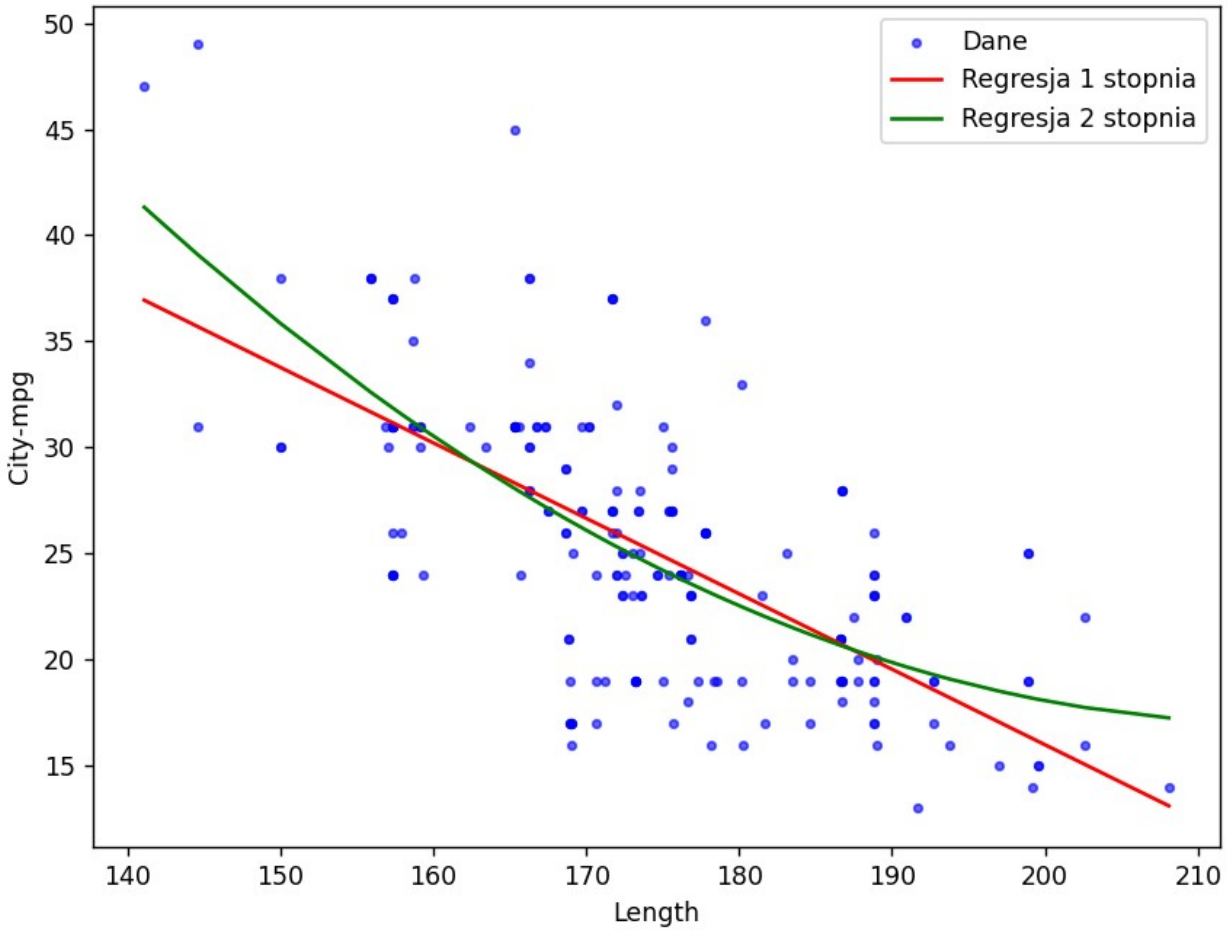
- 7.

```
corr, _ = scs.pearsonr(x, y)
print(f'\ncity-mpg a length: {corr}')
```

```
city-mpg a length: -0.6709086615585711
```

- **8.**

```
plt.figure(figsize=(8, 6))
plt.scatter(x, y, label="Dane", color="blue", alpha=0.6, s=10)
plt.plot( *args: sorted_x, pred_1, label="Regresja 1 stopnia", color="red")
plt.plot( *args: sorted_x, pred_2, label="Regresja 2 stopnia", color="green")
plt.xlabel("Length")
plt.ylabel("City-mpg")
plt.legend()
plt.show()
```



9-10.

```
x_length = df['length'].dropna()

kde_length = scs.gaussian_kde(x_length)

x_vals_length = np.linspace(min(x_length), max(x_length),  num: 1000)
y_vals_length = kde_length(x_vals_length)

_, ax = plt.subplots( nrows: 1,  ncols: 2, figsize=(14, 6))

ax[0].scatter(x_length, np.zeros_like(x_length), label="Próbki (length)", color="blue", alpha=0.6, s=10)
ax[0].plot(x_vals_length, y_vals_length, label="Estymator (length)", color="red", linewidth=2)
ax[0].set_title("Estymator funkcji gęstości: Length")
ax[0].set_xlabel("Length")
ax[0].set_ylabel("Gęstość prawdopodobieństwa")
ax[0].legend()
```

- **10.**

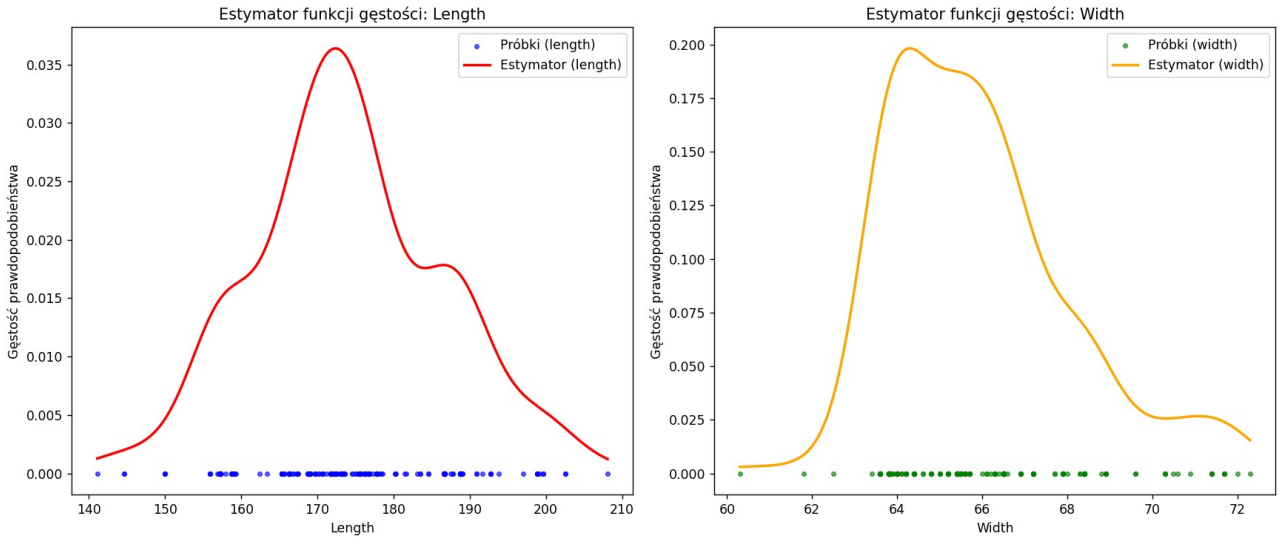
```
x_width = df['width'].dropna()

kde_width = scs.gaussian_kde(x_width)

x_vals_width = np.linspace(min(x_width), max(x_width),  num: 1000)
y_vals_width = kde_width(x_vals_width)

ax[1].scatter(x_width, np.zeros_like(x_width), label="Próbki (width)", color="green", alpha=0.6, s=10)
ax[1].plot(x_vals_width, y_vals_width, label="Estymator (width)", color="orange", linewidth=2)
ax[1].set_title("Estymator funkcji gęstości: Width")
ax[1].set_xlabel("Width")
ax[1].set_ylabel("Gęstość prawdopodobieństwa")
ax[1].legend()

plt.tight_layout()
plt.show()
```



11.

```
x_length = df['length'].dropna()
x_width = df['width'].dropna()

data = np.vstack([x_length, x_width])

kde_2d = scs.gaussian_kde(data)

x_vals = np.linspace(min(x_length), max(x_length),  num: 100)
y_vals = np.linspace(min(x_width), max(x_width),  num: 100)

x,y = np.meshgrid( *xi: x_vals, y_vals)
positions = np.vstack([x.ravel(), y.ravel()])
z = kde_2d(positions).reshape(x.shape)

plt.figure(figsize=(8, 6))
plt.contour( *args: x,y,z, levels=10, cmap="Blues")
plt.plot( *args: x_length, x_width, 'r.', label="Próbki", alpha=0.5)
plt.title("Dwuwymiarowy estymator funkcji gęstości")
plt.xlabel("Length")
plt.ylabel("Width")
plt.legend()
plt.savefig( *args: "png_plot.png", dpi=300)
plt.savefig("pdf_plot.pdf")
plt.show()
```