Adam_Grabowski_WYKRESY

May 24, 2020

[1]:
```
import xlrd
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

[2]:
```
data = pd.read_excel("C:/Users/Adam/muzea.xlsx")
print(data.head())
```

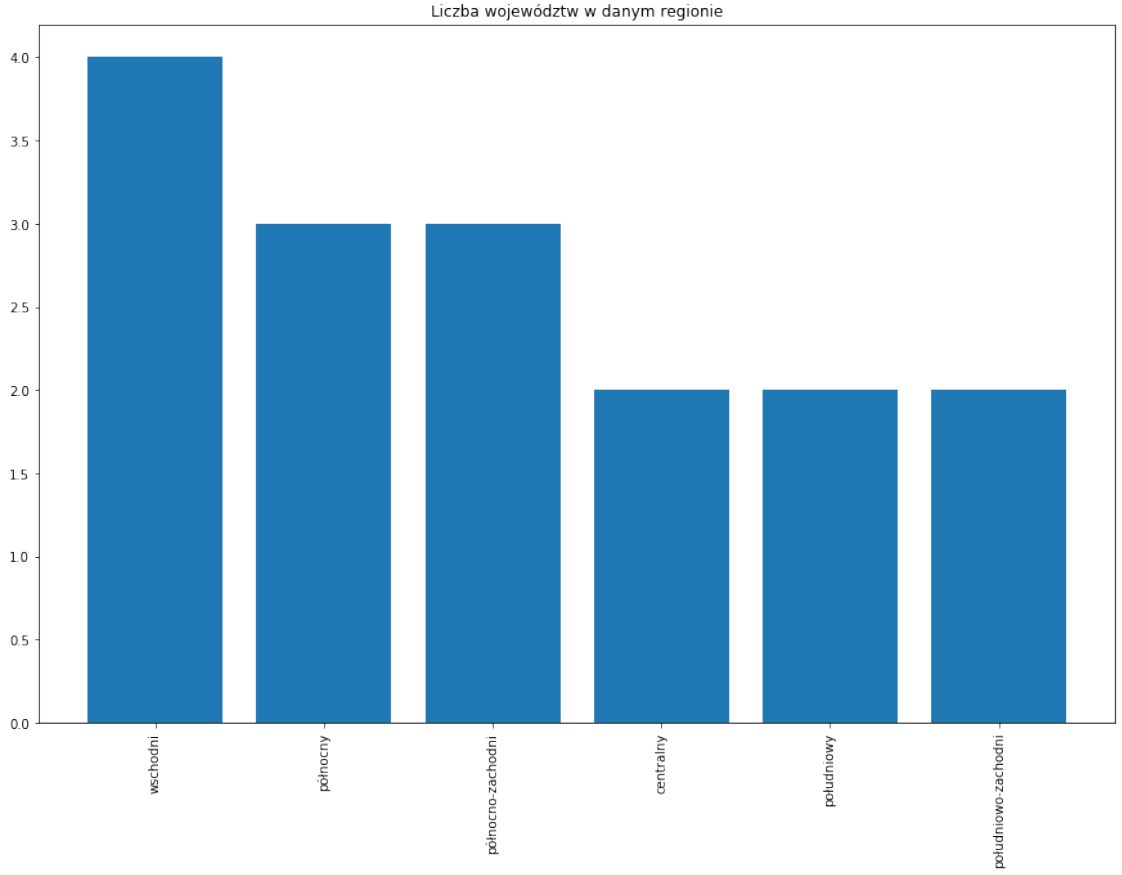
```
                Nazwa  Liczba_zwiedzajacych  ile_mieszkancow  rozwody  \
0        DOLNOŚLĄSKIE                1978088          2904207     5797
1  KUJAWSKO-POMORSKIE                1189942          2086210     4000
2           LUBELSKIE                1371707          2139726     2483
3            LUBUSKIE                 253151          1018075     1937
4             ŁÓDZKIE                 988873          2493603     4267

   separacje  uklad_krazenia  nowotwory  uklad_oddech  samoboj_na_10tys  \
0        175            47.7       25.2           3.7               1.8
1        163            44.9       28.6           6.3               1.5
2        150            51.5       21.2           4.7               2.2
3         56            45.2       25.9           4.4               2.2
4        263            45.5       23.0           5.0               1.9

                Region
0  południowo-zachodni
1             północny
2             wschodni
3    północno-zachodni
4            centralny
```

[3]:
```
# SZEREG ROZDZIELCZY

type = data["Region"]

freq = type.value_counts()
print(freq)
```

```
wschodni              4
północny              3
```

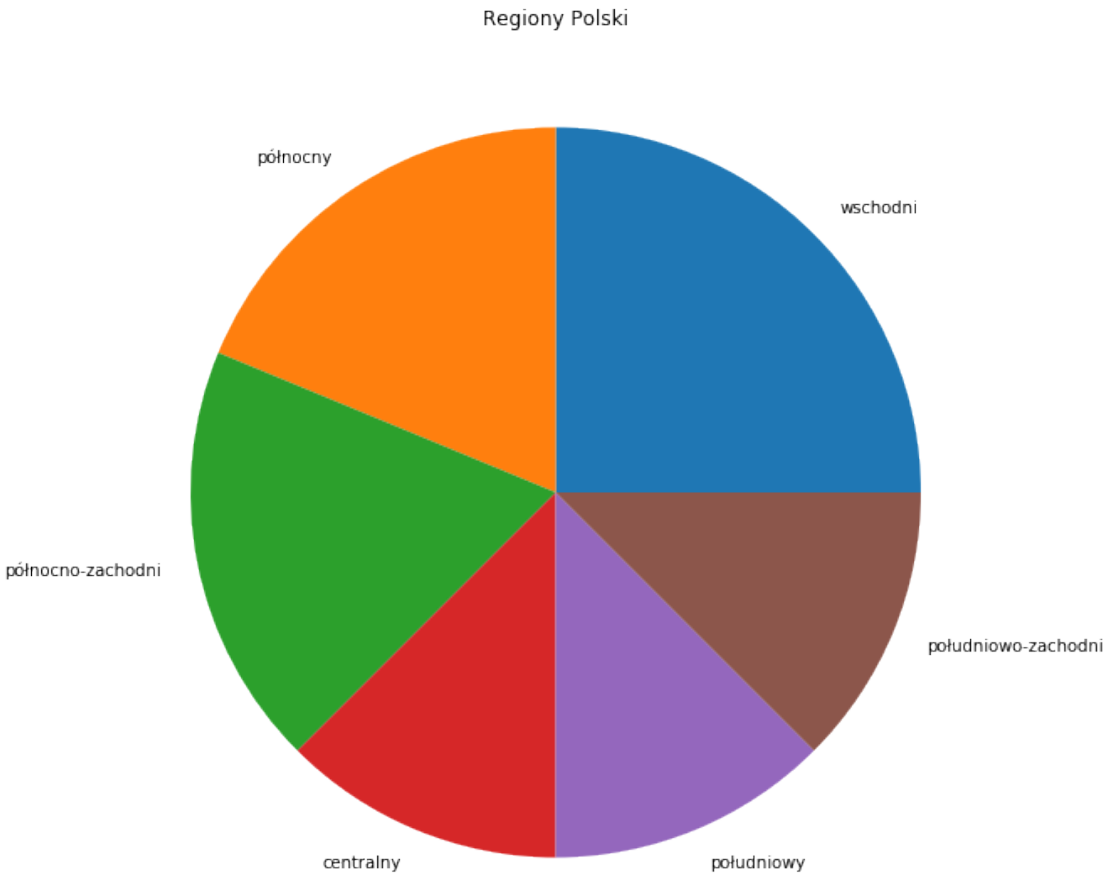
```
północno-zachodni      3
centralny              2
południowy             2
południowo-zachodni    2
Name: Region, dtype: int64
```

[4]:
```
# Wykres słupkowy

x=np.arange(0,len(freq))
plt.figure(figsize=(15,10))
plt.bar(x,freq)
plt.xticks(x,freq.index, rotation=90)
plt.title("Liczba województw w danym regionie")
plt.show()
```



[5]:
```
# Wykres kołowy

plt.figure(figsize=(15,10))
plt.pie(freq, labels=freq.index)
plt.title("Regiony Polski")
```

```
plt.show()
```



[6]:
```
# Przygotowanie danych do analizy ststystycznej

lzm = data["Liczba_zwiedzajacych"]
rzw = data["rozwody"]
sep = data["separacje"]
ukr = data["uklad_krazenia"]
ntw = data["nowotwory"]
uod = data["uklad_oddech"]
sbj = data["samoboj_na_10tys"]
lm = data["ile_mieszkancow"]
```

[7]:
```
# Średnia i odch. standrdowe 2 zmiennych:

print('Liczba ludności w poszcz. wojewodztwach: średnia = %.2f,'
      ' odch. standardowe = %.2f'% (np.mean(lm), np.std(lm)))
```

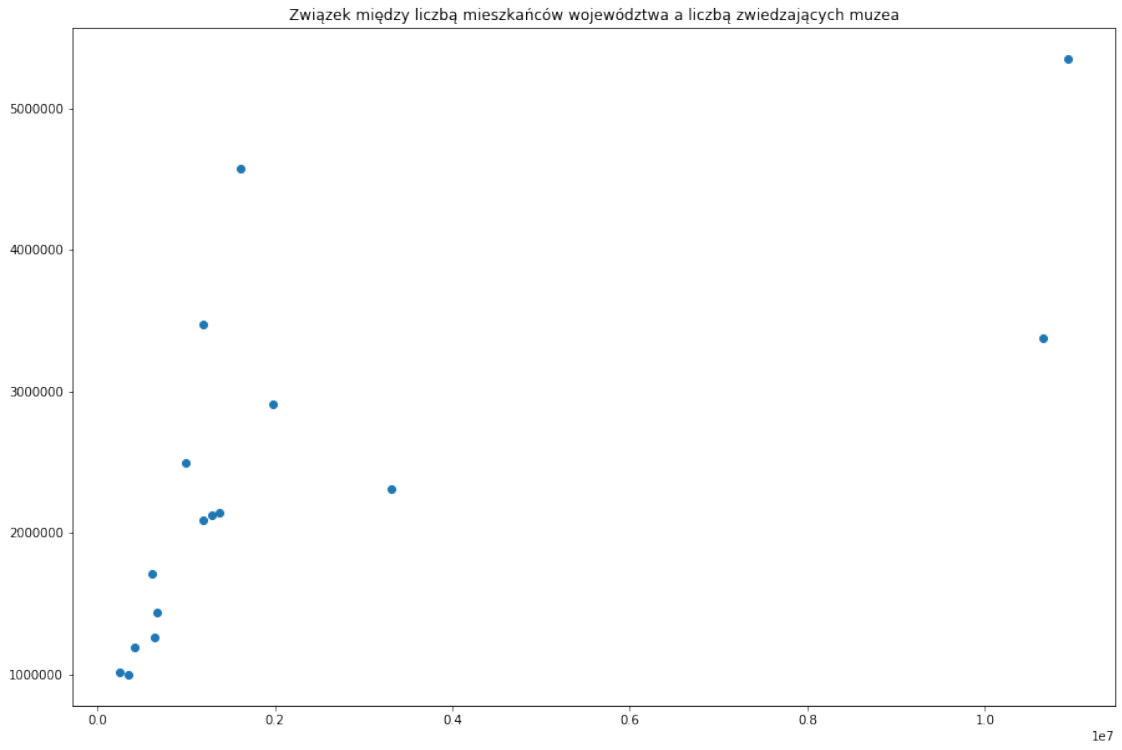
```
print('Liczba zwiedzających muzea w poszcz. wojewodztwach: średnia = %.2f,'
      ' odch. standardowe = %.2f'% (np.mean(lzm), np.std(lzm)))
```

```
Liczba ludności w poszcz. wojewodztwach: średnia = 2402327.44, odch. standardowe
= 1226671.43
Liczba zwiedzających muzea w poszcz. wojewodztwach: średnia = 2343943.00, odch.
standardowe = 3276708.70
```

[8]:
```
# KORELACJA (r-Pearsona)

# Czy liczba zwiedzających muzea w poszczególnych województwach jest skorelowana␣
 ↪z liczbą mieszkańców?

from scipy.stats import pearsonr
corr, _ = pearsonr(lzm, lm)
print(corr)
```

```
0.6820735005588521
```

[9]:
```
# Wykres ilustrujący ten związek

plt.figure(figsize=(15,10))
plt.scatter(lzm, lm)
plt.title("Związek między liczbą mieszkańców województwa a liczbą zwiedzających␣
 ↪muzea")
plt.show()
```



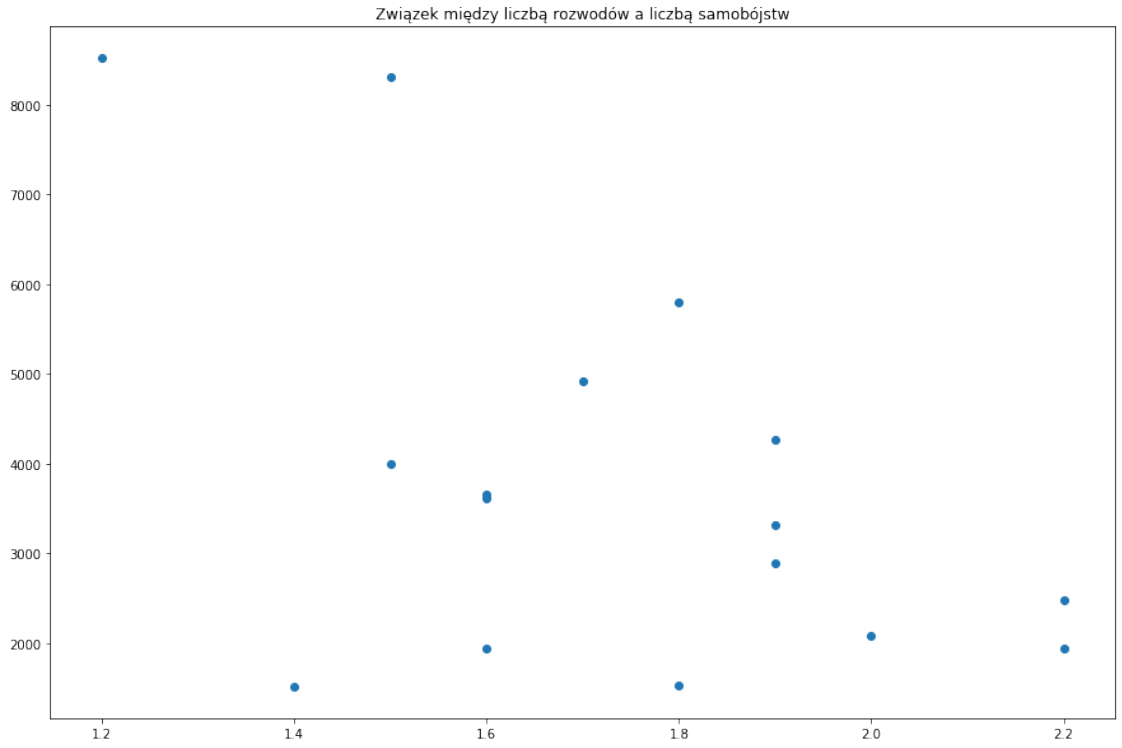
[10]:
```
# Czy liczba samobójstw w poszczególnych województwach
# jest skorelowana z liczbą rozwodów?

print('Liczba samobójstw na 10000 mieszkańców'
      ' w poszcz. wojewodztwach: średnia = %.2f,'
      ' odch. standardowe = %.2f'% (np.mean(sbj), np.std(sbj)))

print('Liczba rozwodów w poszcz. wojewodztwach: średnia = %.2f,'
      ' odch. standardowe = %.2f'% (np.mean(rzw), np.std(rzw)))
```

```
Liczba samobójstw na 10000 mieszkańców w poszcz. wojewodztwach: średnia = 1.74,
odch. standardowe = 0.27
Liczba rozwodów w poszcz. wojewodztwach: średnia = 3801.06, odch. standardowe =
2110.09
```

[11]:
```
corr, _ = pearsonr(sbj, rzw)
print(corr)
```

```
-0.5193122415497085
```

[12]:
```
# Wykres ilustrujący ten związek

plt.figure(figsize=(15,10))
plt.scatter(sbj, rzw)
plt.title("Związek między liczbą rozwodów"
          " a liczbą samobójstw")
plt.show()
```



[13]:
```
# Czy liczba rozwodów w poszczególnych województwach
# jest skorelowana z liczbą separacji?

print('Liczba separacji w poszcz. wojewodztwach: średnia = %.2f,'
      ' odch. standardowe = %.2f'% (np.mean(sep), np.std(sep)))
print('Liczba rozwodów w poszcz. wojewodztwach: średnia = %.2f,'
      ' odch. standardowe = %.2f'% (np.mean(rzw), np.std(rzw)))
```

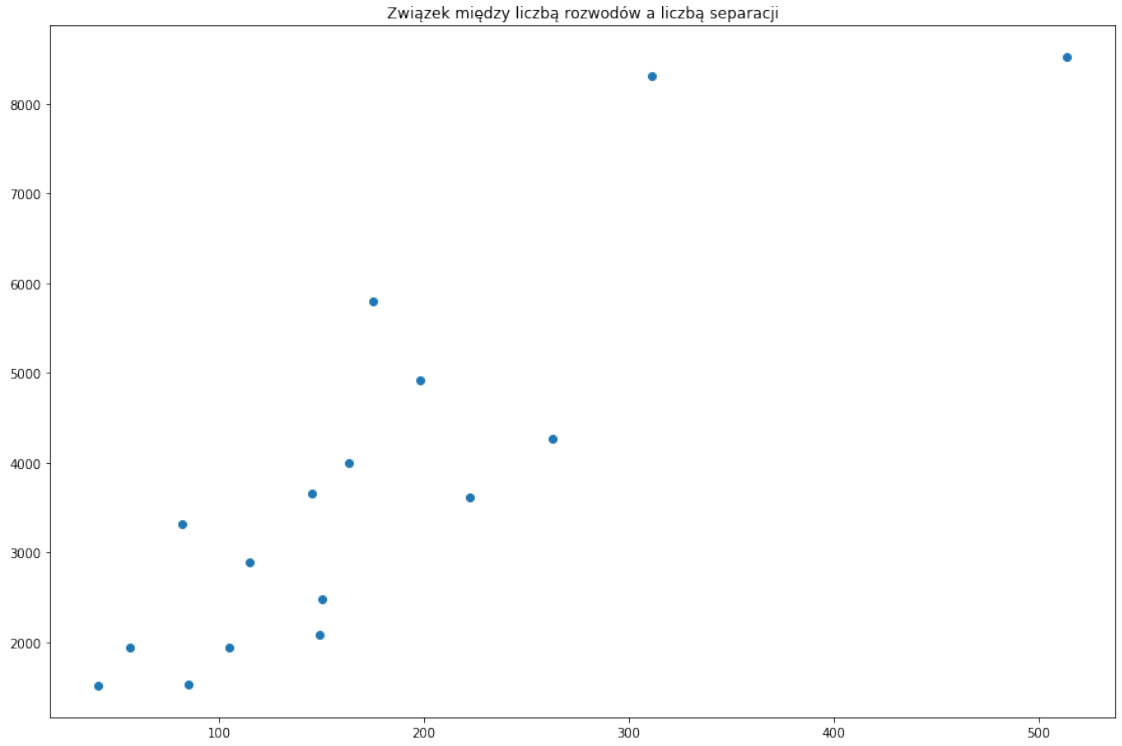
```
Liczba separacji w poszcz. wojewodztwach: średnia = 173.38, odch. standardowe =
112.80
Liczba rozwodów w poszcz. wojewodztwach: średnia = 3801.06, odch. standardowe =
2110.09
```

[14]:
```
corr, _ = pearsonr(sep, rzw)
print(corr)
```

```
0.8701540026702343
```

[15]:
```
# Wykres ilustrujący ten związek

plt.figure(figsize=(15,10))
plt.scatter(sep, rzw)
plt.title("Związek między liczbą rozwodów"
```

```
          " a liczbą separacji")
plt.show()
```

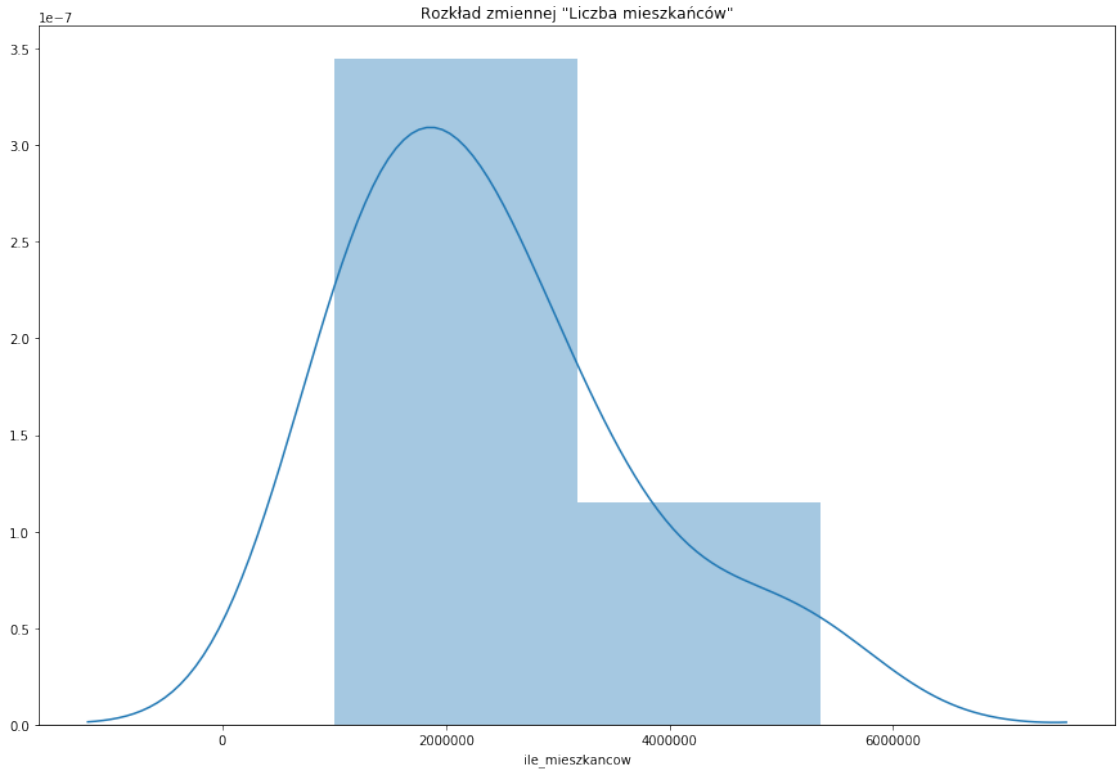


[16]:
```
# REGRESJA LINIOWA

# W jakim stopniu takie czynniki, jak liczba zgonów
# (z powodu róznych rodzajów chorób oraz na skutek
# samobójstwa) oraz separacje i rozwody wpływają
# na liczbę mieszkańców.

import seaborn as seabornInstance
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn import metrics
%matplotlib inline
```

[17]:
```
# Podział danych na (potencjalne) wyznaczniki (X) oraz zmienną przewidywaną (y)

X = data[['uklad_krazenia', 'nowotwory', 'uklad_oddech',
          'samoboj_na_10tys', 'rozwody', 'separacje']].values
y = data['ile_mieszkancow'].values
```

```
# Jaki rozkład ma zmienna y:

plt.figure(figsize=(15,10))
plt.tight_layout()
plt.title('Rozkład zmiennej "Liczba mieszkańców"')
seabornInstance.distplot(lm)
```

[17]: <matplotlib.axes._subplots.AxesSubplot at 0x24dd1eeea88>



[18]:
```
# Dane rozdzielamy na zestaw treningowy (80%) i testowy (20%)

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=0)
```

[19]:
```
# Model poddajemy treningowi:

regressor = LinearRegression()
regressor.fit(X_train, y_train)
```

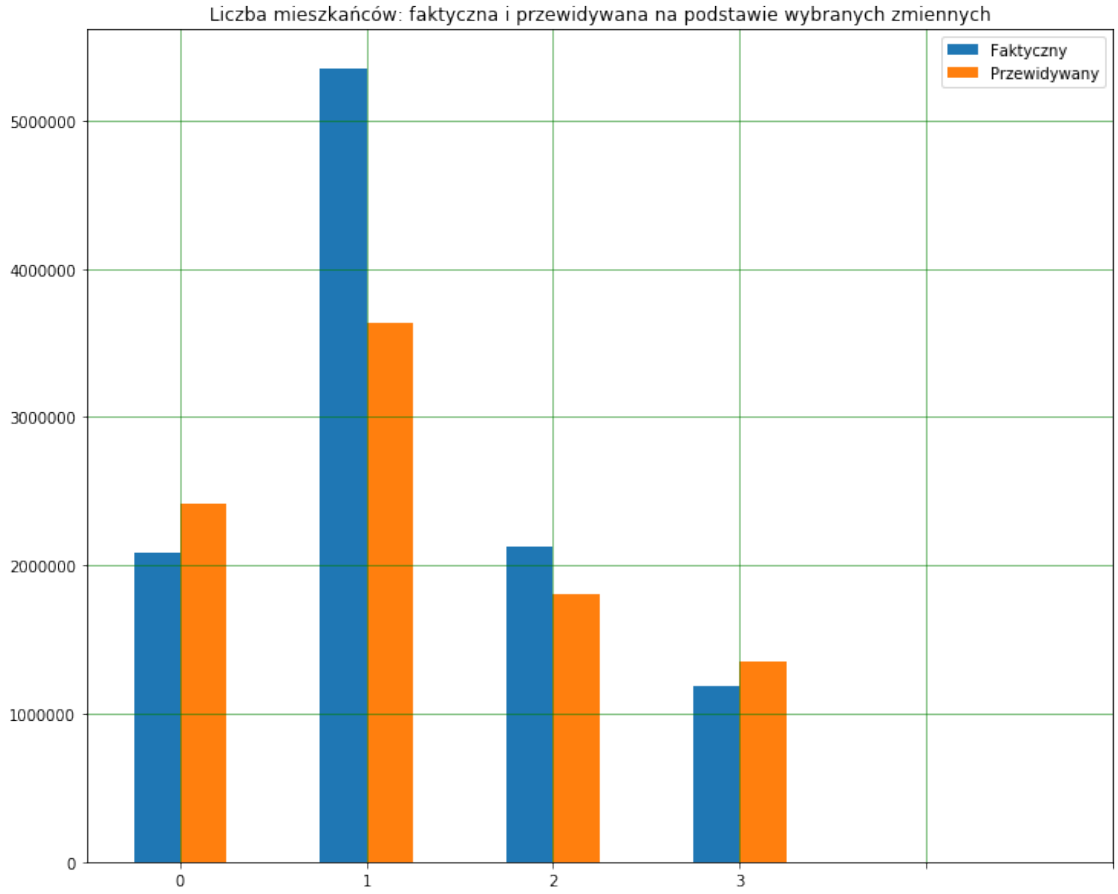
[19]: LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)

[20]:
```
# Współczynniki regresji ustalone przez model:

# coeff_df = pd.DataFrame(regressor.coef_, X.columns, columns=['Coefficient'])
# coeff_df

# Ten kod niestety nie działa, jeszcze nie wiem, dlaczego.
```

[21]:
```
# Model dokonuje predykcji:

y_pred = regressor.predict(X_test)
```

[22]:
```
# Porównujemy wartości faktyczne i przewidywane przez model

df = pd.DataFrame({'Faktyczny': y_test, 'Przewidywany': y_pred})
df1 = df.head(6)
```

[23]:
```
df1.plot(kind='bar',figsize=(12,10))
plt.grid(which='major', linestyle='-', linewidth='0.5', color='green')
plt.grid(which='minor', linestyle=':', linewidth='0.5', color='black')
plt.xticks(x,rotation=0)
plt.title("Liczba mieszkańców: faktyczna i przewidywana na podstawie wybranych␣
 ↪zmiennych")
plt.show()
```


Liczba mieszkańców: faktyczna i przewidywana na podstawie wybranych zmiennych
Faktyczny
Przewidywany
5000000
4000000
3000000
2000000
1000000
0
0
1
2
3

[ ]:

[ ]: