pd2_mikolaj_spytek

March 22, 2021

[1]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
from category_encoders import TargetEncoder
from category_encoders import OneHotEncoder
from category_encoders import CountEncoder
from category_encoders import OrdinalEncoder
import random
from math import floor
from sklearn.metrics import mean_squared_error
from sklearn.impute import KNNImputer
```

1 Praca domowa 2

Mikołaj Spytek

W tej pracy domowej zajmuję się zbiorem danych Allegro. W pierwszej części chodzi o kodowanie zmiennych kategorycznych, natomiast w drugiej o preprocessing - uzupełnianie danych brakujących.

1.1 Część pierwsza

W pierwszej części kodujemy zmienne kategoryczne - zmienną `it_location` za pomocą target encoding oraz zmienną `main_category` za pomocą one-hot encodingu oraz dwóch innych metod.

[2]:
```
# wczytanie zbioru danych
df = pd.read_csv("https://www.dropbox.com/s/360xhh2d9lnaek3/
 ↪allegro-api-transactions.csv?dl=1")
# zauważyłem, ze nazwy lokalizacji są różnie pisane, więc ujednolicam
df[["it_location"]] = df[["it_location"]].apply(lambda x: x.str.lower())

#stworzenie kopii, aby każdy typ encodingu był na osobnej ramce
dftarget = df.copy()
dfonehot = df.copy()
dfcount = df.copy()
dfordinal = df.copy()
df_part2 = df[["price", "it_seller_rating", "it_quantity"]]
```

[3]:
```
df[["it_location"]].describe()
```

[3]:

	it_location
count	420020
unique	7903
top	warszawa
freq	27042

Gdy spojrzymy na zmienną, którą chcemy zakodować, zauważamy, że przyjmuje ona 7903 różnych wartości. Gdybyśmy więc chieli zastosować one-hot encoding, to powstałoby właśnie tyle nowych kolumn, czyli zmiennych. Ponieważ mamy dużo obserwacji - nadal byłoby około rząd wielkości więcej niż zmiennych, to mogłoby się okazać, że model będzie działać, lecz target encoding jest sposobem na zmniejszenie ilości zmiennych objaśniających.

[4]:
```
# target encoding zmiennej it_location
en = TargetEncoder()
dftarget["target_encoded_it_location"] = en.
 ↪fit_transform(dftarget["it_location"], dftarget["price"])
# sprawdzenie czy pojawiła się nowa kolumna
dftarget.head()
```

```
c:\users\mikołaj\appdata\local\programs\python\python38\lib\site-
packages\category_encoders\utils.py:21: FutureWarning: is_categorical is
deprecated and will be removed in a future version.  Use is_categorical_dtype
instead
  elif pd.api.types.is_categorical(cols):
```

[4]:

	lp	date	item_id	categories	pay_option_on_delivery	pay_option_transfer	seller	price	it_is_allegro_standard
0	0	2016-04-03 21:21:08	4753602474	['Komputery', 'Dyski i napędy', 'Nośniki', 'No…	1	1	radzioch666	59.99	1
1	1	2016-04-03 15:35:26	4773181874	['Odzież, Obuwie, Dodatki', 'Bielizna damska',…	1	1	InwestycjeNET	4.90	1
2	2	2016-04-03 14:14:31	4781627074	['Dom i Ogród', 'Budownictwo i Akcesoria', 'Śc…	1	1	otostyl_com	109.90	1
3	3	2016-04-03 19:55:44	4783971474	['Książki i Komiksy', 'Poradniki i albumy', 'Z…	1	1	Matfel1	18.50	0
4	4	2016-04-03 18:05:54	4787908274	['Odzież, Obuwie, Dodatki', 'Ślub i wesele', '…	1	1	PPHU_RICO	19.90	1

```
   it_quantity  it_is_brand_zone  it_seller_rating          it_location  \
0          997                 0             50177            warszawa
1         9288                 0             12428            warszawa
2          895                 0              7389              leszno
3          971                 0             15006  wola krzysztoporska
4          950                 0             32975           białystok

             main_category  target_encoded_it_location
0                Komputery                   84.132898
1  Odzież, Obuwie, Dodatki                   84.132898
2              Dom i Ogród                   64.883187
3        Książki i Komiksy                   35.433365
4  Odzież, Obuwie, Dodatki                   73.772916
```

Widzimy, że w ramce danych pojawiła się nowa kolumna: `target_encoded_it_location`. Jest to zmienna zawierająca zakodowane wartości kategorii.

Target encoding polega na pogrupowaniu danych według kategorii, a następnie wyliczenia dla każdej z tych kategorii średniej wartości zmiennej wyjaśnianej. Otrzymaną w ten sposób wartością kodujemy wszystkie obserwacje z danej kategorii.

Na przykładzie można sprawdzić, czy ten Encoder rzeczywiście tak działa. Dla Warszawy policzymy średnią cenę produktu "ręcznie", a następnie sprawdzimy czy taka sama wartość została nadana przez Encoder.

[5]:
```
print("Średnia wyliczona ręcznie: ", dftarget.
 ↪loc[dftarget["it_location"]=="warszawa", "price"].mean())
print("Wartość z encodingu: ", dftarget.
 ↪loc[dftarget["it_location"]=="warszawa", "target_encoded_it_location"].
 ↪head(1))
```

```
Średnia wyliczona ręcznie:  84.13289808446122
Wartość z encodingu:  0    84.132898
Name: target_encoded_it_location, dtype: float64
```

Wygląda na to, że wszystko się zgadza.

Jakie są wady takiego kodowania?

- zamieniając zmienną kategoryczną na tylko jedną zmienną numeryczną wprowadzamy porządek (możliwość porównania) kategorii, który wcześniej nie istniał,
- jeśli w którejś kategorii było mało obserwacji, to średnia wartość targetu może być w łatwy sposób zaburzona,
- dane zakodowane w ten sposób mogą powodować przeuczenie się modelu

[6]:
```
df[["main_category"]].describe()
```

[6]:
```
       main_category
count         420020
unique            27
```

```
top      Dom i Ogród
freq           91042
```

Widzimy, że ta zmienna przyjmuje już tylko 27 unikalnych wartości, więc zastosowanie one-hot encodingu ma tu dużo większy sens.

[7]:
```
en = OneHotEncoder(use_cat_names=True)

dfonehot = dfonehot.join(en.fit_transform(dfonehot["main_category"]))
dfonehot.head()
```

```
c:\users\mikołaj\appdata\local\programs\python\python38\lib\site-
packages\category_encoders\utils.py:21: FutureWarning: is_categorical is
deprecated and will be removed in a future version.  Use is_categorical_dtype
instead
  elif pd.api.types.is_categorical(cols):
```

[7]:

	lp	date	item_id	\
0	0	2016-04-03 21:21:08	4753602474	
1	1	2016-04-03 15:35:26	4773181874	
2	2	2016-04-03 14:14:31	4781627074	
3	3	2016-04-03 19:55:44	4783971474	
4	4	2016-04-03 18:05:54	4787908274	

	categories	pay_option_on_delivery	\
0	['Komputery', 'Dyski i napędy', 'Nośniki', 'No…	1	
1	['Odzież, Obuwie, Dodatki', 'Bielizna damska',…	1	
2	['Dom i Ogród', 'Budownictwo i Akcesoria', 'Śc…	1	
3	['Książki i Komiksy', 'Poradniki i albumy', 'Z…	1	
4	['Odzież, Obuwie, Dodatki', 'Ślub i wesele', '…	1	

	pay_option_transfer	seller	price	it_is_allegro_standard	\
0	1	radzioch666	59.99	1	
1	1	InwestycjeNET	4.90	1	
2	1	otostyl_com	109.90	1	
3	1	Matfel1	18.50	0	
4	1	PPHU_RICO	19.90	1	

	it_quantity	…	main_category_Filmy	main_category_Fotografia	\
0	997	…	0	0	
1	9288	…	0	0	
2	895	…	0	0	
3	971	…	0	0	
4	950	…	0	0	

	main_category_Biuro i Reklama	main_category_Instrumenty	\
0	0	0	
1	0	0	

```
2                              0                              0
3                              0                              0
4                              0                              0

   main_category_Muzyka  main_category_Konsole i automaty  \
0                     0                                 0
1                     0                                 0
2                     0                                 0
3                     0                                 0
4                     0                                 0

   main_category_Sprzęt estradowy, studyjny i DJ-ski  \
0                                                  0
1                                                  0
2                                                  0
3                                                  0
4                                                  0

   main_category_Antyki i Sztuka  main_category_Bilety  \
0                              0                     0
1                              0                     0
2                              0                     0
3                              0                     0
4                              0                     0

   main_category_Nieruchomości
0                            0
1                            0
2                            0
3                            0
4                            0

[5 rows x 41 columns]
```

Widzimy, że teraz ramka danych ma 41, kolumny, podczas gdy wcześniej miała 14. Oznacza to, że OneHotEncoder dodał 27 nowych zmiennych - dokładnie tyle ile było unikalnych wartości kolumny `main_category`. Jest to zarówno zaleta, jak i wada tego rodzaju kodowania. Dzięki temu, że każda kategoria ma osobną zmienną, możemy mieć pewność, że nie będą na siebie wpływały w modelu, tak jak to może być w przypadku kodowań, które tworzą tylko jedną zmienną.

[8]:
```
en = CountEncoder()

dfcount["count_encoded_main_category"] = en.
 ↪fit_transform(dfcount["main_category"])

dfcount.head()
```

[8]:
```
   lp                 date     item_id  \
0   0  2016-04-03 21:21:08  4753602474
1   1  2016-04-03 15:35:26  4773181874
2   2  2016-04-03 14:14:31  4781627074
3   3  2016-04-03 19:55:44  4783971474
4   4  2016-04-03 18:05:54  4787908274

                                        categories  pay_option_on_delivery  \
0  ['Komputery', 'Dyski i napędy', 'Nośniki', 'No…                       1
1  ['Odzież, Obuwie, Dodatki', 'Bielizna damska',…                       1
2  ['Dom i Ogród', 'Budownictwo i Akcesoria', 'Śc…                       1
3  ['Książki i Komiksy', 'Poradniki i albumy', 'Z…                       1
4  ['Odzież, Obuwie, Dodatki', 'Ślub i wesele', '…                       1

   pay_option_transfer         seller   price  it_is_allegro_standard  \
0                    1    radzioch666   59.99                       1
1                    1  InwestycjeNET    4.90                       1
2                    1    otostyl_com  109.90                       1
3                    1        Matfel1   18.50                       0
4                    1      PPHU_RICO   19.90                       1

   it_quantity  it_is_brand_zone  it_seller_rating          it_location  \
0          997                 0             50177             warszawa
1         9288                 0             12428             warszawa
2          895                 0              7389               leszno
3          971                 0             15006  wola krzysztoporska
4          950                 0             32975            białystok

             main_category  count_encoded_main_category
0                Komputery                        14491
1  Odzież, Obuwie, Dodatki                        54257
2              Dom i Ogród                        91042
3        Książki i Komiksy                        11572
4  Odzież, Obuwie, Dodatki                        54257
```

W wyniku zastosowania tego kodowania, każdej obserwacji przypisywana jest liczba produktów w danej kategorii. Mamy więc tylko jedną dodatkową kolumnę. W ramach sprawdzenia można popatrzeć na wiersz 2 z powyższej ramki - obserwacja z kategorii `Dom i ogród` została zakodowana jako 91042, a wcześniej gdy sprawdzaliśmy liczbę kategorii, polecenie `df["main_category"].describe()` pokazało, że najczęstszą kategorią jest właśnie ta i pojawia się 91042 razy w tym zbiorze danych.

Wadą tego typu encodingu również jest to, że kategoriom nadajemy arbitralny porządek. Dodatkowo kategorie o podobnej liczności będą miały podobną wartość zmiennej kodującej kategorie, a mogą być one zupełnie różny wpływ na zmienną wyjaśnianą.

[9]:
```
en = OrdinalEncoder()
```

```
dfordinal["ordinal_encoded_main_category"] = en.
 ↪fit_transform(dfordinal["main_category"])

dfordinal.head()
```

[9]:

```
   lp                 date     item_id  \
0   0  2016-04-03 21:21:08  4753602474
1   1  2016-04-03 15:35:26  4773181874
2   2  2016-04-03 14:14:31  4781627074
3   3  2016-04-03 19:55:44  4783971474
4   4  2016-04-03 18:05:54  4787908274

                                          categories  pay_option_on_delivery  \
0  ['Komputery', 'Dyski i napędy', 'Nośniki', 'No…                       1
1  ['Odzież, Obuwie, Dodatki', 'Bielizna damska',…                       1
2  ['Dom i Ogród', 'Budownictwo i Akcesoria', 'Śc…                       1
3  ['Książki i Komiksy', 'Poradniki i albumy', 'Z…                       1
4  ['Odzież, Obuwie, Dodatki', 'Ślub i wesele', '…                       1

   pay_option_transfer         seller   price  it_is_allegro_standard  \
0                    1    radzioch666   59.99                       1
1                    1  InwestycjeNET    4.90                       1
2                    1    otostyl_com  109.90                       1
3                    1        Matfel1   18.50                       0
4                    1      PPHU_RICO   19.90                       1

   it_quantity  it_is_brand_zone  it_seller_rating          it_location  \
0          997                 0             50177             warszawa
1         9288                 0             12428             warszawa
2          895                 0              7389               leszno
3          971                 0             15006  wola krzysztoporska
4          950                 0             32975            białystok

             main_category  ordinal_encoded_main_category
0                Komputery                              1
1  Odzież, Obuwie, Dodatki                              2
2              Dom i Ogród                              3
3        Książki i Komiksy                              4
4  Odzież, Obuwie, Dodatki                              2
```

Ta metoda jest bardzo naiwna, i często pociąga niepożądane konsekwencje. Enkoder po prostu przyporządkowuje kategoriom kolejne liczby naturalne. Znów problem pojawia się, ponieważ dostajemy jakiś porządek na kategoriach, którego wcześniej nie było. Model może więc się nauczyć nieistniejących zależności.

1.2 Część druga

Wypełnianie braków w zmiennych.

[10]:
```
df_part2.head()
```

[10]:

	price	it_seller_rating	it_quantity
0	59.99	50177	997
1	4.90	12428	9288
2	109.90	7389	895
3	18.50	15006	971
4	19.90	32975	950

[11]:
```
# ustawiamy seed, aby wyniki były powtarzalne
random.seed(123)
# wybieramy podzbiór kolumn
df_part2 = df[["price", "it_seller_rating", "it_quantity"]]

imputer = KNNImputer(weights="uniform", n_neighbors=5)

#ograniczmy liczbę rekordów - przy pełnej ramce obliczenia wykonywały się␣
 ↪bardzo długo
df_part2 = df_part2[:floor(len(df_part2)*0.1)]
rmse = [None for i in range(10)]

type(df_part2)
len(df_part2)
for i in range(10):
    # robimy kopię ramki
    df_part2copy = df_part2.copy()
    # wybieramy 10% losowych indeksów
    toberemoved = random.sample(range(0, len(df_part2)), floor(len(df_part2)*0.
 ↪1))
    # usuwamy
    df_part2copy.iloc[toberemoved, 1] = None
    # imputacja
    df_part2copy = pd.DataFrame(imputer.fit_transform(df_part2copy),␣
 ↪columns=["price", "it_seller_rating", "it_quantity"])

    # wyliczenie błędu
    rmse[i] = np.sqrt(mean_squared_error(df_part2["it_seller_rating"],␣
 ↪df_part2copy["it_seller_rating"]))
    print("RMSE {:.2f}".format(rmse[i]))


print("Odchylenie standardowe miary RMSE w 10 próbach wynosi: {:.2f}".format(np.
 ↪std(rmse)))
```

```
RMSE 10589.80
RMSE 10703.49
RMSE 9976.69
RMSE 11261.98
```

```
RMSE 11508.35
RMSE 10826.57
RMSE 11105.03
RMSE 11205.60
RMSE 10841.34
RMSE 10512.52
Odchylenie standardowe miary RMSE w 10 próbach wynosi: 419.63
```

Powtarzamy ten sam eksperyment tylko, że usuwamy wartości z dwóch kolumn

[12]:
```
df_part2 = df[["price", "it_seller_rating", "it_quantity"]]

imputer = KNNImputer(weights="uniform", n_neighbors=5)

#ograniczmy liczbę rekordów
df_part2 = df_part2[:floor(len(df_part2)*0.1)]
rmse1 = [None for i in range(10)]
rmse2 = [None for i in range(10)]


type(df_part2)
len(df_part2)
for i in range(10):
    df_part2copy = df_part2.copy()
    toberemoved = random.sample(range(0, len(df_part2)), floor(len(df_part2)*0.
 ↪1))
    toberemoved2 = random.sample(range(0, len(df_part2)), floor(len(df_part2)*0.
 ↪1))
    df_part2copy.iloc[toberemoved, 1] = None
    df_part2copy.iloc[toberemoved, 2] = None


    df_part2copy = pd.DataFrame(imputer.fit_transform(df_part2copy),␣
 ↪columns=["price", "it_seller_rating", "it_quantity"])


    rmse1[i] = np.sqrt(mean_squared_error(df_part2["it_seller_rating"],␣
 ↪df_part2copy["it_seller_rating"]))
    rmse2[i] = np.sqrt(mean_squared_error(df_part2["it_quantity"],␣
 ↪df_part2copy["it_quantity"]))

    print("RMSE kolumny it_seller_rating: {:.2f}".format(rmse1[i]))
    print("RMSE kolumny it_quantity: {:.2f}".format(rmse2[i]))
```

```
#print("Odchylenie standardowe miary RMSE kolumny it_seller_rating w 10 próbach␣
 ↪wynosi: {:.2f}".format(np.std(rmse1)))
#print("Odchylenie standardowe miary RMSE kolumny it_quantity w 10 próbach␣
 ↪wynosi: {:.2f}".format(np.std(rmse2)))

d = {"RMSE kolumny it_seller_rating": rmse1, "Rmse kolumny it_quantity":rmse2 }
results = pd.DataFrame(d)
results
```

```
RMSE kolumny it_seller_rating: 12138.33
RMSE kolumny it_quantity: 7904.22
RMSE kolumny it_seller_rating: 12439.86
RMSE kolumny it_quantity: 8032.27
RMSE kolumny it_seller_rating: 12252.89
RMSE kolumny it_quantity: 7801.74
RMSE kolumny it_seller_rating: 12847.25
RMSE kolumny it_quantity: 8267.00
RMSE kolumny it_seller_rating: 12038.04
RMSE kolumny it_quantity: 7816.94
RMSE kolumny it_seller_rating: 12483.09
RMSE kolumny it_quantity: 7904.61
RMSE kolumny it_seller_rating: 12620.75
RMSE kolumny it_quantity: 7597.03
RMSE kolumny it_seller_rating: 12467.79
RMSE kolumny it_quantity: 8313.81
RMSE kolumny it_seller_rating: 12420.58
RMSE kolumny it_quantity: 7827.02
RMSE kolumny it_seller_rating: 11961.49
RMSE kolumny it_quantity: 7996.07
```

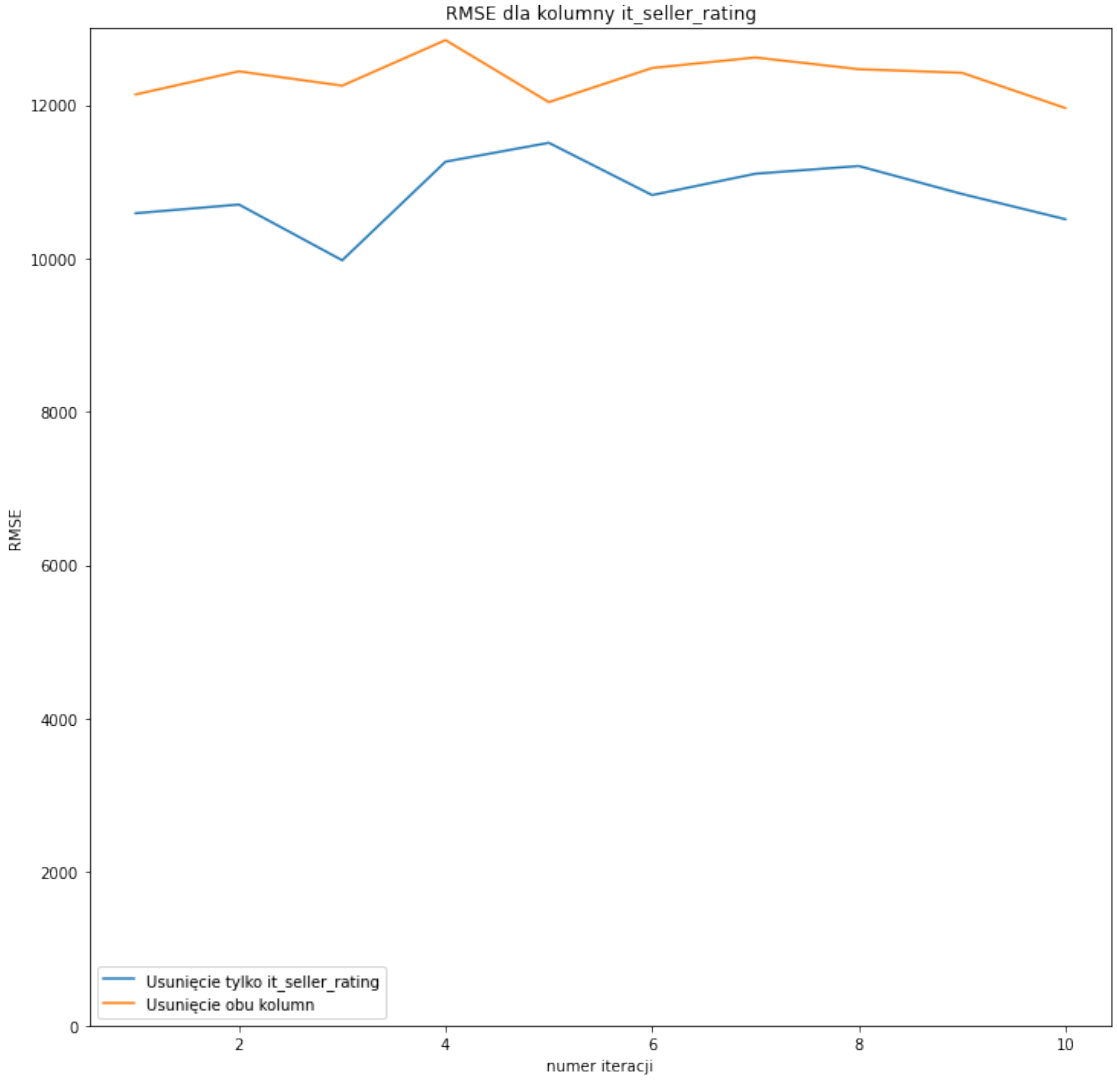
[12]:

	RMSE kolumny it_seller_rating	Rmse kolumny it_quantity
0	12138.327332	7904.218393
1	12439.861217	8032.271669
2	12252.893293	7801.736286
3	12847.251121	8267.001091
4	12038.036251	7816.939468
5	12483.093094	7904.606089
6	12620.747872	7597.028461
7	12467.790074	8313.811728
8	12420.584858	7827.016226
9	11961.488033	7996.065046

[13]:
```
np.std(results)
```

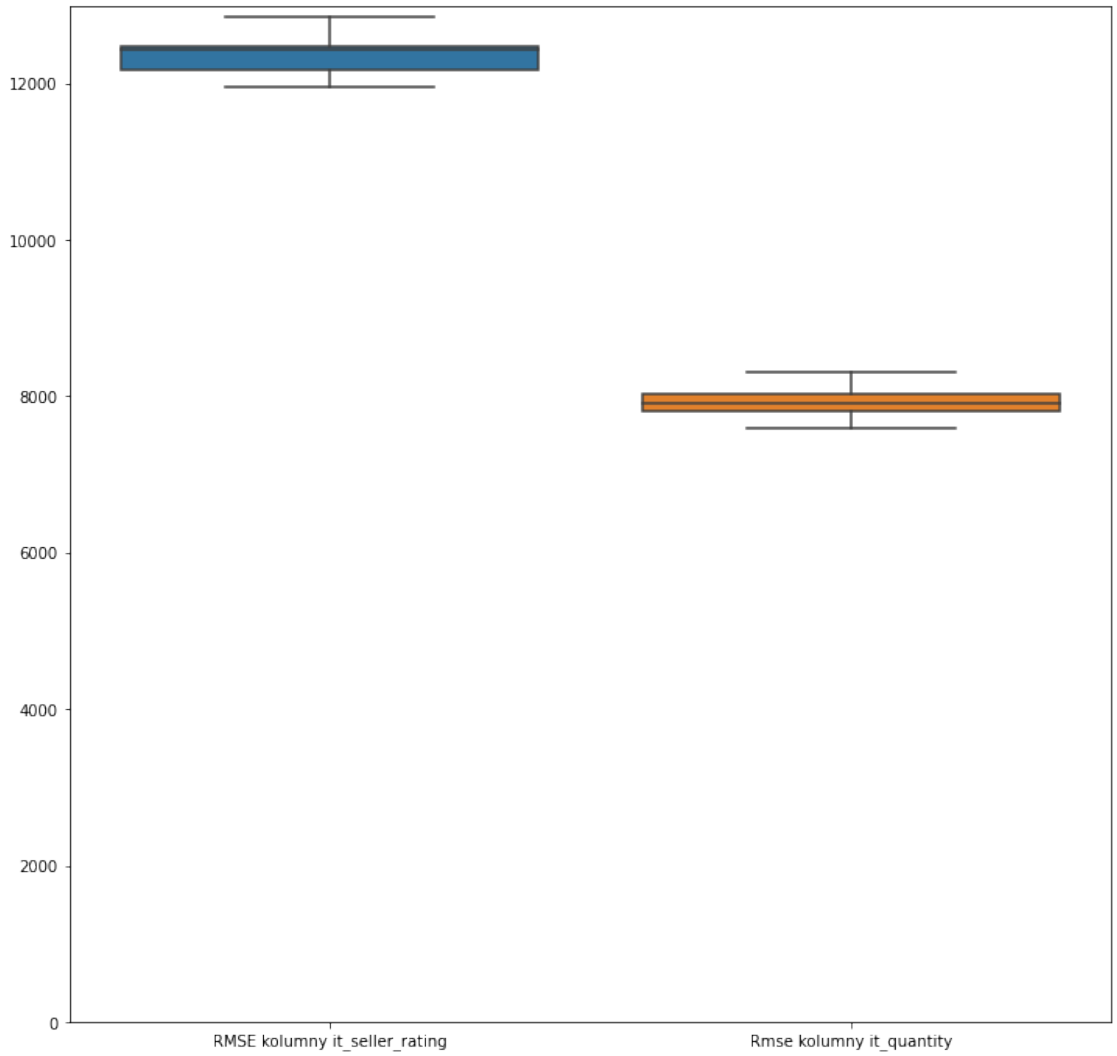
[13]:
```
RMSE kolumny it_seller_rating    257.839891
Rmse kolumny it_quantity         206.189840
dtype: float64
```

[16]:
```
fig = plt.figure(figsize=(12,12))
p = plt.plot([i for i in range(1, 11)], rmse, [i for i in range(1, 11)], rmse1)
plt.ylim([0, 13000])
plt.legend(["Usunięcie tylko it_seller_rating", "Usunięcie obu kolumn"])
plt.title("RMSE dla kolumny it_seller_rating")
plt.xlabel("numer iteracji")
plt.ylabel("RMSE")
plt.show()
```



[19]:
```
fig = plt.figure(figsize=(12,12))
sns.boxplot(data=results)
plt.ylim([0,13000])
```

[19]: (0.0, 13000.0)



Z powyższych wykresów widać, że jeśli usuniemy dane z więcej niż jednej kolumny, algorytm imputacji działa gorzej. Może to być spowodowane, że w pierwszym przypadku znajduje "lepszych", tzn. bardziej powiązanych sąsiadów.

Innym czynnikiem, który może mieć wpływ na wynik, jest użyta metryka. W tym przypadku używam domyślnej metryki euklidesowej.