In [49]:
```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import seaborn as sns

from category_encoders import TargetEncoder, OneHotEncoder, BinaryEncoder, CountEncc
from sklearn.impute import KNNImputer
from sklearn.metrics import mean_squared_error
import random
```

In [48]:
```
df = pd.read_csv('https://www.dropbox.com/s/360xhh2d9lnaek3/allegro-api-transactions
df.head(10)
```

Out[48]:

	lp	date	item_id	categories	pay_option_on_delivery	pay_option_transfer	seller
0	0	2016-04-03 21:21:08	4753602474	['Komputery', 'Dyski i napędy', 'Nośniki', 'No...	1	1	radzioch666
1	1	2016-04-03 15:35:26	4773181874	['Odzież, Obuwie, Dodatki', 'Bielizna damska',...	1	1	InwestycjeNET
2	2	2016-04-03 14:14:31	4781627074	['Dom i Ogród', 'Budownictwo i Akcesoria', 'Śc...	1	1	otostyl_com
3	3	2016-04-03 19:55:44	4783971474	['Książki i Komiksy', 'Poradniki i albumy', 'Z...	1	1	Matfel1
4	4	2016-04-03 18:05:54	4787908274	['Odzież, Obuwie, Dodatki', 'Ślub i wesele', '...	1	1	PPHU_RICO
5	5	2016-04-03 16:31:01	4790991674	['Odzież, Obuwie, Dodatki', 'Bielizna damska',...	1	1	Noemi-bielizna
6	6	2016-04-03 17:56:11	4790991674	['Odzież, Obuwie, Dodatki', 'Bielizna damska',...	1	1	Noemi-bielizna
7	7	2016-04-03 11:58:55	4824025074	['Biżuteria i Zegarki', 'Zegarki', 'Dziecięce']	1	1	handel_barbo
8	8	2016-04-03 18:18:37	4826332874	['RTV i AGD', 'AGD drobne', 'Higiena i pielęgn...	1	1	jupiter2009
9	9	2016-04-03 22:56:56	4828603874	['RTV i AGD', 'Kamery', 'Zasilanie', 'Zasilacze']	1	1	e-trade-com-pl

Patrząc na kilka pierwszych rekordów z wczytanej ramki danych, możemy zauważyć, że wielkość liter w nazwach lokacji jest różna. Ujednolićmy więc dane nazwy zmieniając ich pisownię na wielkie litery.

In [31]:
```
df[["it_location"]] = df[["it_location"]].apply(lambda x: x.str.upper())
```

1. Kodowanie zmiennych kategorycznych

Target encoding

In [32]:
```
df_target_encoding = df.copy()
df_target_encoding[["it_location"]].describe()
```

Out[32]:

	it_location
count	420020
unique	7903
top	WARSZAWA
freq	27042

In [33]:
```
encoder = TargetEncoder()
df_target_encoding["it_location_target_encoded"] = encoder.fit_transform(
    df_target_encoding["it_location"], df_target_encoding["price"])
df_target_encoding[["it_location_target_encoded"]].describe()
```

```
c:\users\artur\appdata\local\programs\python\python39\lib\site-packages\category_e
ncoders\utils.py:21: FutureWarning: is_categorical is deprecated and will be remov
ed in a future version.  Use is_categorical_dtype instead
  elif pd.api.types.is_categorical(cols):
```

Out[33]:

	it_location_target_encoded
count	420020.000000
mean	75.707902
std	85.588009
min	1.001266
25%	49.402415
50%	68.376518
75%	84.132898
max	16016.652061

In [34]:
```
len(df_target_encoding[["it_location_target_encoded"]].value_counts())
```

Out[34]: 5140

Możemy zauważyć, że zmienna it_location przyjmuje aż 7903 róznych wartości. Przy zastosowaniu one hot encodingu wynikowa ramka danych miałaby 7903 nowych kolumn, co zdecydowanie nie byłoby dobrym wyjściem. Możemy zauważyć, że po zastosowaniu target encodingu liczba unikatowych wartości zmniejszyła się do 5140 (w wynikowej, zakodowanej kolumnie).

Wartości w nowo powstałej kolumnie są wyliczoną średnią wartością zmiennej docelowej cechy dla każdej z kategorii.

Target encoding jednak może prowadzić do przeuczenia modelu (jest powiązany ze zmienną objaśnialną). Wprowadza także możliwość porównywania zakodowanych w ten sposób kategorii, co w naszym przypadku wydaje się nie mieć dużego sensu.

Kodowanie main_category

In [37]:
```
df[['main_category']].describe()
```

Out[37]:

	main_category
count	420020
unique	27
top	Dom i Ogród
freq	91042

One Hot Encoder

In [42]:
```
oh_encoder = OneHotEncoder(use_cat_names=True).fit_transform(df["main_category"])
oh_encoder.head()
```

```
c:\users\artur\appdata\local\programs\python\python39\lib\site-packages\category_e
ncoders\utils.py:21: FutureWarning: is_categorical is deprecated and will be remov
ed in a future version.  Use is_categorical_dtype instead
  elif pd.api.types.is_categorical(cols):
```

Out[42]:

	main_category_Komputery	main_category_Odzież, Obuwie, Dodatki	main_category_Dom i Ogród	main_category_Książki i Komiksy	main
0	1	0	0	0	
1	0	1	0	0	
2	0	0	1	0	
3	0	0	0	1	
4	0	1	0	0	

5 rows × 27 columns

Powstało 27 nowych kolumn, każda dla unikalnej wartości zmiennej main_category.

Count Encoder

In [44]:
```
count_encoder = CountEncoder().fit_transform(df["main_category"])
count_encoder.head()
```

Out[44]:

	main_category
0	14491
1	54257
2	91042
3	11572
4	54257

Kodowanie count encoder tworzy nam jedną kolumnę, gdzie dla każdej obserwacji przypisana jest liczba produktów z odpowiadającej kategorii.

Binary Encoder

In [45]:
```
binary_encoder = BinaryEncoder().fit_transform(df["main_category"])
binary_encoder.head()
```

```
c:\users\artur\appdata\local\programs\python\python39\lib\site-packages\category_e
ncoders\utils.py:21: FutureWarning: is_categorical is deprecated and will be remov
ed in a future version.  Use is_categorical_dtype instead
  elif pd.api.types.is_categorical(cols):
```

Out[45]:

	main_category_0	main_category_1	main_category_2	main_category_3	main_category_4	main_categ
0	0	0	0	0	0	
1	0	0	0	0	1	
2	0	0	0	0	1	
3	0	0	0	1	0	
4	0	0	0	0	1	

Kodowanie binarne jest bardzo podobne do One Hote'a, ale przechowuje kategorie w postaci binarnych bitstringów.

2. Uzupełnianie braków

In [90]:
```
df2 = df[['price','it_seller_rating','it_quantity']].copy()
df2 = df2.sample(frac=0.1)
df2.head()
```

Out[90]:

	price	it_seller_rating	it_quantity
394646	24.00	49199	4
8117	2.89	517	9360
222269	12.00	190	1
312221	5.99	1850	992
133141	78.70	133555	1768

In [92]:
```
rmse_errors = []

for i in range(10):
    df2_1 = df2.copy()
    remove_index = df2_1.sample(frac=0.1).index
    df2_1.loc[remove_index, 'it_seller_rating'] = None
    df2_1 = pd.DataFrame(KNNImputer(weights='uniform', n_neighbors=5).fit_transform(
    error = np.sqrt(mean_squared_error(df2["it_seller_rating"], df2_1["it_seller_rat
    rmse_errors.append(error)
    print(error)
```

```
12355.446463293469
11864.261184446112
11668.855088283568
11649.03097237245
12044.556506561277
12158.873752937656
11488.778850382967
11888.563110786572
11374.554809774649
11388.90835671248
```

In [95]:
```
np.std(rmse_errors)
```

Out[95]:
```
315.3640286073554
```

In [97]:
```
rmse_errors1 = []
rmse_errors2 = []

for i in range(10):
    df2_2 = df2.copy()
    remove_index1 = df2_2.sample(frac=0.1).index
    remove_index2 = df2_2.sample(frac=0.1).index

    df2_2.loc[remove_index1, 'it_seller_rating'] = None
    df2_2.loc[remove_index2, 'it_quantity'] = None

    df2_2 = pd.DataFrame(KNNImputer(weights='uniform', n_neighbors=5).fit_transform(
    error1 = np.sqrt(mean_squared_error(df2["it_seller_rating"], df2_2["it_seller_ra
    error2 = np.sqrt(mean_squared_error(df2["it_quantity"], df2_2["it_quantity"]))

    rmse_errors1.append(error1)
    rmse_errors2.append(error2)
    print("RMSE it_seller_rating - ", error1)
    print("RMSE it_quantity - ", error2)
```

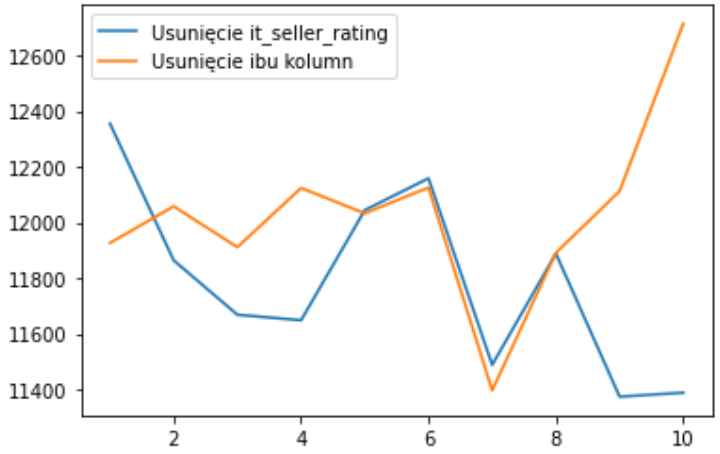
```
RMSE it_seller_rating -  11926.5159178538
RMSE it_quantity -  7945.340641031782
RMSE it_seller_rating -  12058.240393017715
RMSE it_quantity -  7839.645395447648
RMSE it_seller_rating -  11911.45740986274
RMSE it_quantity -  7624.71484154093
RMSE it_seller_rating -  12124.118489389131
RMSE it_quantity -  7898.356029193538
RMSE it_seller_rating -  12033.020157623941
RMSE it_quantity -  7704.1602258328885
RMSE it_seller_rating -  12125.751846972124
RMSE it_quantity -  8129.220098891593
RMSE it_seller_rating -  11396.940751299491
RMSE it_quantity -  7777.622204977856
RMSE it_seller_rating -  11891.159333737094
RMSE it_quantity -  7754.557793748457
RMSE it_seller_rating -  12112.874127837395
RMSE it_quantity -  7810.737653049214
RMSE it_seller_rating -  12714.306939483808
RMSE it_quantity -  7861.411340216268
```

In [98]:
```
print("RMSE it_seller_rating std - ",np.std(rmse_errors1))
print("RMSE it_quantity std - ",np.std(rmse_errors2))
```

```
RMSE it_seller_rating std -  306.3945736107649
RMSE it_quantity std -  132.34899771256127
```

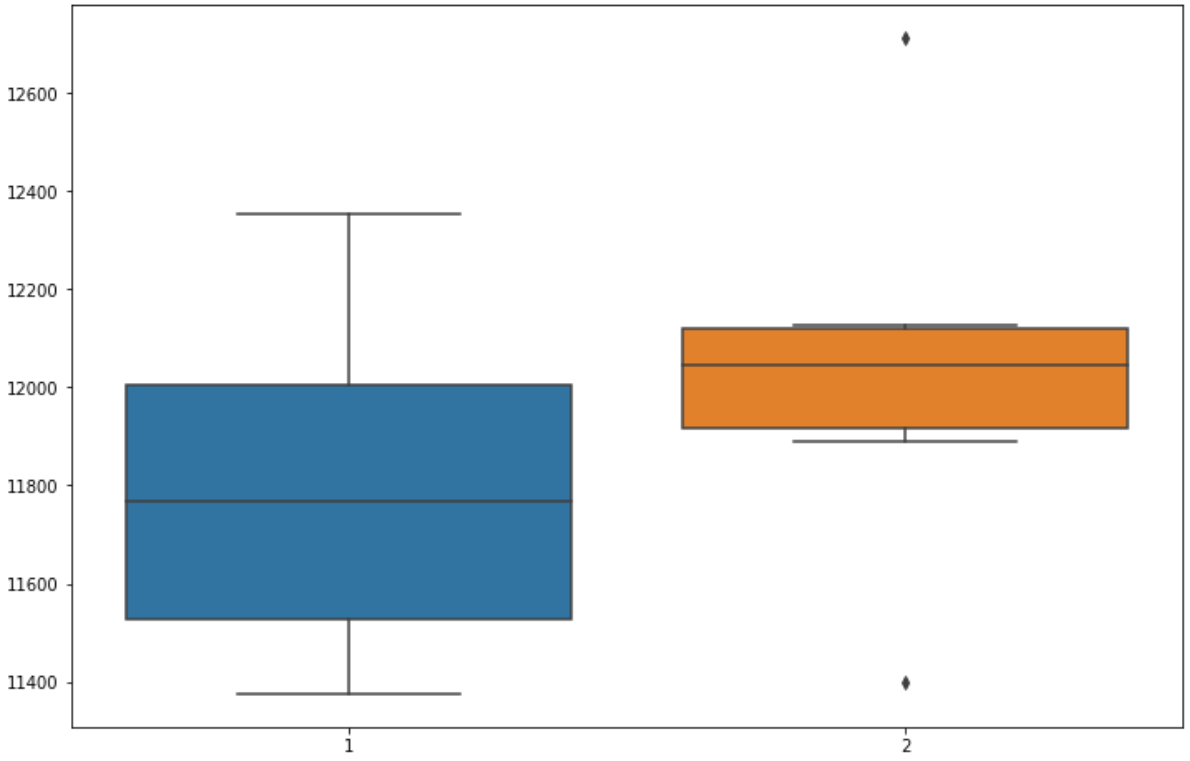
In [102]:
```
plt.plot(range(1,11), rmse_errors, range(1,11), rmse_errors1)
plt.legend(['Usunięcie it_seller_rating', 'Usunięcie obu kolumn'])
```

Out[102]:
```
<matplotlib.legend.Legend at 0x1e0fe358100>
```



In [106]:
```
plotdf = pd.DataFrame({"1": rmse_errors, "2":rmse_errors1})
fig, ax = plt.subplots(figsize=(12,8))
sns.boxplot(data=plotdf)
```

Out[106]:
```
<AxesSubplot:>
```



Na wykresach możemy zaobserwować, że przy usunięciu danych z dwóch kolumn imputacja będzie obarczona większym błędem, choć nadal losowość w doborze usuwanych danych może powodować pewne odstępstwa.