PD2-Przemyslaw-Olender

March 23, 2021

[1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from category_encoders import TargetEncoder, OneHotEncoder, CountEncoder,␣
↪OrdinalEncoder
import random
from sklearn.impute import KNNImputer
from sklearn.metrics import mean_squared_error as rmse
from statistics import stdev
```

[2]:
```
allegro = pd.read_csv('allegro-api-transactions.csv')

allegro.head()
```

[2]:
```
   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
0                 Komputery
1  Odzież, Obuwie, Dodatki
2               Dom i Ogród
3         Książki i Komiksy
4  Odzież, Obuwie, Dodatki
```

[3]:
```
allegro.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 420020 entries, 0 to 420019
Data columns (total 14 columns):
 #   Column                  Non-Null Count   Dtype
---  ------                  --------------   -----
 0   lp                      420020 non-null  int64
 1   date                    420020 non-null  object
 2   item_id                 420020 non-null  int64
 3   categories              420020 non-null  object
 4   pay_option_on_delivery  420020 non-null  int64
 5   pay_option_transfer     420020 non-null  int64
 6   seller                  420020 non-null  object
 7   price                   420020 non-null  float64
 8   it_is_allegro_standard  420020 non-null  int64
 9   it_quantity             420020 non-null  int64
 10  it_is_brand_zone        420020 non-null  int64
 11  it_seller_rating        420020 non-null  int64
 12  it_location             420020 non-null  object
 13  main_category           420020 non-null  object
dtypes: float64(1), int64(8), object(5)
memory usage: 36.9+ MB
```

0.1 Zadanie 1

Wykonaj target encoding dla zmiennej it_location. Czy i jakie są przewagi target encoding nad one-hot? Jako target traktujemy kolumnę price (będzie to więc zadanie regresji).

Zastosuj trzy metody encodingu (one-hot + "dwie nowe") dla kolumny main_category. "Nowe metody" proszę wybrać spośród wymienionych na stronie https://contrib.scikit-learn.org/category_encoders/. W przypadku, gdy użyta metoda nie działa proszę o stosowną adnotację. Opisz wyniki.

Zwizalizuj wynik oraz wyjaśnij czym się różnią sposoby kodowania (czemu to działa).

[4]:
```
location = allegro[['it_location']].groupby(['it_location']).size().
 ↪sort_values(ascending=False).reset_index()
location.columns = ['location', 'count']

location.loc[location['location'].apply(lambda x:'warszawa' in x.lower())]
```

[4]:

	location	count
0	Warszawa	23244
19	WARSZAWA	2672
48	warszawa	1123
182	Warszawa/ Sprzedaż tylko wysyłkowa	333
416	Warszawa, Katowice	146
…	…	…
9842	Warszawa Skierniewice Grójec Płock	1
9843	Warszawa Stara Miłosna	1
9858	RADOM WARSZAWA LUBLIN	1
9885	RASZYN WARSZAWA	1
9929	Płock/Warszawa	1

[386 rows x 2 columns]

Warszawa, WARSZAWA i warszawa to zupełnie to samo, można ujednolicić, trzeba się zastanowić nad innymi nazwami.

Tak samo jest z innymi miastami, żeby ujednolicić najłatwiej przekształcić wszystkie nazwy tak, żeby składały się tylko z małych liter, nie będzie wtedy problemu z wieloczłonowymi nazwami.

[5]:
```
allegro['it_location'] = allegro['it_location'].apply(lambda x: x.lower())
```

[6]:
```
allegro['it_location'].value_counts()
```

[6]:
```
warszawa                    27042
kraków                      16581
łódź                        12433
poznań                      11197
internet                    10992
                            …
gałków mały                     1
zielona góa                     1
uszew                           1
wrocław / mokronos dolny        1
koszwały                        1
Name: it_location, Length: 7903, dtype: int64
```

[7]:
```
target_encoder = TargetEncoder()

allegro['target_encoding_location'] =  target_encoder.
 ↪fit_transform(allegro['it_location'], allegro['price'])
```

```
c:\users\user\appdata\local\programs\python\python38-32\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):
```

[8]:
```
allegro.head()
```

[8]:

	lp	date	item_id	categories	pay_option_on_delivery	pay_option_transfer	seller	price	it_is_allegro_standard	it_quantity	it_is_brand_zone	it_seller_rating	it_location	main_category	target_encoding_location
0	0	2016-04-03 21:21:08	4753602474	['Komputery', 'Dyski i napędy', 'Nośniki', 'No…	1	1	radzioch666	59.99	1	997	0	50177	warszawa	Komputery	84.132898
1	1	2016-04-03 15:35:26	4773181874	['Odzież, Obuwie, Dodatki', 'Bielizna damska',…	1	1	InwestycjeNET	4.90	1	9288	0	12428	warszawa	Odzież, Obuwie, Dodatki	84.132898
2	2	2016-04-03 14:14:31	4781627074	['Dom i Ogród', 'Budownictwo i Akcesoria', 'Śc…	1	1	otostyl_com	109.90	1	895	0	7389	leszno	Dom i Ogród	64.883187
3	3	2016-04-03 19:55:44	4783971474	['Książki i Komiksy', 'Poradniki i albumy', 'Z…	1	1	Matfel1	18.50	0	971	0	15006	wola krzysztoporska	Książki i Komiksy	35.433365
4	4	2016-04-03 18:05:54	4787908274	['Odzież, Obuwie, Dodatki', 'Ślub i wesele', '…	1	1	PPHU_RICO	19.90	1	950	0	32975	białystok	Odzież, Obuwie, Dodatki	73.772916

Pojawiła się nowa kolumna - ‘target_encoding_location’, zakodowana wartość to średnia z wartośći kolumny ‘price’, dla danej wartości ‘it_location’.

[9]:
```
try:
    OneHotEncoder(use_cat_names=True, cols = ['it_location']).
 ↪fit_transform(allegro.it_location)
except MemoryError as error:
    print('Brakuje pamieci, ', error)
```

```
c:\users\user\appdata\local\programs\python\python38-32\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):
```

```
Brakuje pamieci,  Unable to allocate 477. MiB for an array with shape (7903,
7905) and data type int64
```

OneHotEncoding potrzbuje dużo dodatkowej pamięci do zakodowania zmiennej 'it_location', chcąc dołączyć zakodowaną macierz do ramki danych potrzebował by jeszcze znacznie więcej zasobów - to spory minus, kolejna pamięć byłaby potrzeba na wykonywanie operacji na nowo powstałej ramce.

W tym przypadku TargetEncoder sprawdza się znacznie lepiej, lecz on też ma swoje wady. Kodując za pomocą średniej stwarza możliwość porównywania zakodowanych wartości, co może być mylące.

0.1.1 OneHotEncoder

Kodowanie kolumny 'main_category'

[10]:
```
OneHotEncoder(cols = ['main_category'], use_cat_names=True).
 ↪fit_transform(allegro.main_category)
```

```
c:\users\user\appdata\local\programs\python\python38-32\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):
```

[10]:

	main_category_Komputery	main_category_Odzież, Obuwie, Dodatki	\
0	1	0	
1	0	1	
2	0	0	
3	0	0	
4	0	1	
…	…	…	
420015	0	0	
420016	0	0	
420017	0	1	
420018	0	0	
420019	0	0	

	main_category_Dom i Ogród	main_category_Książki i Komiksy	\

```
0                                0                                0
1                                0                                0
2                                1                                0
3                                0                                1
4                                0                                0
…                              …                                …
420015                           0                                0
420016                           0                                0
420017                           0                                0
420018                           0                                0
420019                           0                                0

        main_category_Biżuteria i Zegarki  main_category_RTV i AGD  \
0                                       0                        0
1                                       0                        0
2                                       0                        0
3                                       0                        0
4                                       0                        0
…                                     …                        …
420015                                  0                        1
420016                                  0                        0
420017                                  0                        0
420018                                  0                        0
420019                                  0                        0

        main_category_Motoryzacja  main_category_Dla Dzieci  \
0                               0                          0
1                               0                          0
2                               0                          0
3                               0                          0
4                               0                          0
…                             …                          …
420015                          0                          0
420016                          0                          0
420017                          0                          0
420018                          0                          1
420019                          1                          0

        main_category_Uroda  main_category_Telefony i Akcesoria  …  \
0                         0                                   0  …
1                         0                                   0  …
2                         0                                   0  …
3                         0                                   0  …
4                         0                                   0  …
…                       …                                   …  …
420015                    0                                   0  …
420016                    1                                   0  …
```

```
420017                    0                                    0  …
420018                    0                                    0  …
420019                    0                                    0  …

        main_category_Filmy  main_category_Fotografia  \
0                         0                         0
1                         0                         0
2                         0                         0
3                         0                         0
4                         0                         0
…                       …                         …
420015                    0                         0
420016                    0                         0
420017                    0                         0
420018                    0                         0
420019                    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
…                                 …                        …
420015                              0                          0
420016                              0                          0
420017                              0                          0
420018                              0                          0
420019                              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
…                        …                                 …
420015                     0                                 0
420016                     0                                 0
420017                     0                                 0
420018                     0                                 0
420019                     0                                 0

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

```
3                                                    0
4                                                    0
…                                                  …
420015                                               0
420016                                               0
420017                                               0
420018                                               0
420019                                               0

        main_category_Antyki i Sztuka  main_category_Bilety  \
0                                   0                     0
1                                   0                     0
2                                   0                     0
3                                   0                     0
4                                   0                     0
…                                 …                     …
420015                              0                     0
420016                              0                     0
420017                              0                     0
420018                              0                     0
420019                              0                     0

        main_category_Nieruchomości
0                                 0
1                                 0
2                                 0
3                                 0
4                                 0
…                               …
420015                            0
420016                            0
420017                            0
420018                            0
420019                            0

[420020 rows x 27 columns]
```

0.1.2 CountEncoder

Każdej wartości z kolumny 'main_categoty' przyporządkowuje liczbę jej wystąpień.

[11]:
```
CountEncoder(cols=['main_category']).fit_transform(allegro.main_category)
```

```
c:\users\user\appdata\local\programs\python\python38-32\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):
```

[11]:
```
        main_category
0               14491
1               54257
2               91042
3               11572
4               54257
…                   …
420015          20341
420016          28096
420017          54257
420018          42107
420019          45941

[420020 rows x 1 columns]
```

0.1.3 OrdinalEncoder

Tworzy kolumne intów, każdej unikatowj wartości z 'main_category' przyporządkowuje kolejną wartość naturalną.

[12]:
```
OrdinalEncoder(cols=['main_category']).fit_transform(allegro.main_category)
```

```
c:\users\user\appdata\local\programs\python\python38-32\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):
```

[12]:
```
        main_category
0                   1
1                   2
2                   3
3                   4
4                   2
…                   …
420015              6
420016              9
420017              2
420018              8
420019              7

[420020 rows x 1 columns]
```

Tak jak poprzednio, OneHotEncoding potrzebuje najwięcej pamięci i dodaje do ramki 27 nowych kolumn. CountEncoder i OrdinalEncoder zachowują się podobnie do TargetEncoder, Count zamiast średniej liczy liczbę wystąpień, da się porównywać zakodowane zmienne, co nie jest dobrą cechą. Ordinal dopasowuje kolejno dobrane liczby naturalne, co za tym idzie nie mają one nic wspólengo z kodowaną zmienną, również da się je porównywać.

0.2 Zadanie 2

W tej części zadania traktujemy zmienną price nie jak target a zmienną objaśniającą. Zbiór danych ograniczamy do zmiennych numerycznych tj. price, it_seller_rating i it_quantity.

Proszę losowo usunąć 10% wartości ze zmiennej it_seller_rating i je uzupełnić z użyciem jednego z automatycznych narzędzi: Nearest neighbors imputation lub Multivariate feature imputation (https://scikit-learn.org/stable/modules/impute.html). Następnie należy porównać wartości imputowane z oryginalnymi (polecam miarę RMSE). Eksperyment powtórzyć 10 razy i zobaczyć jakie będzie odchylenie standardowe wyniku. Następnie zrobić analogiczną analizę gdy oprócz losowego usuwania 10% wartości z kolumny it_seller_rating usuniemy także losowo 10% ze zmiennej it_quantity. (w przypadku problemów wydajnościowych proszę ograniczyć liczbę rekordów).

Opisać wnioski z analizy jakości imputacji i umieścić podsumowujący wykres.

[13]:
```
allegro_num = allegro[['price', 'it_seller_rating', 'it_quantity']].
 ↪sample(10000).reset_index(drop = True)

#ognaiczyłem dane ze względu na problemy z wydajnością
```

[14]:
```
print(allegro_num.shape)
allegro_num.head()
```

```
(10000, 3)
```

[14]:

	price	it_seller_rating	it_quantity
0	79.00	3942	3
1	25.90	4562	151
2	35.00	19207	8187
3	9.99	12352	1
4	20.00	583	7

[15]:
```
allegro_num.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 3 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   price             10000 non-null  float64
 1   it_seller_rating  10000 non-null  int64
 2   it_quantity       10000 non-null  int64
dtypes: float64(1), int64(2)
memory usage: 234.4 KB
```

[16]:
```
indexes = allegro_num.sample(int(len(allegro_num) * 0.1)).index

lost_data = allegro_num.copy(deep = True)
```

```
lost_data.loc[indexes,'it_seller_rating'] = np.nan
```

[17]:
```
lost_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 3 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   price             10000 non-null  float64
 1   it_seller_rating  9000 non-null   float64
 2   it_quantity       10000 non-null  int64
dtypes: float64(2), int64(1)
memory usage: 234.4 KB
```

[18]:
```
imputer = KNNImputer(n_neighbors = 3,weights = 'uniform').
 ↪fit_transform(lost_data)

filled_data = pd.DataFrame(imputer)

filled_data.columns = ['price', 'it_seller_rating', 'it_quantity']
```

[19]:
```
filled_data.head()
```

[19]:

	price	it_seller_rating	it_quantity
0	79.00	3942.0	3.0
1	25.90	4562.0	151.0
2	35.00	19207.0	8187.0
3	9.99	12352.0	1.0
4	20.00	583.0	7.0

[20]:
```
filled_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 3 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   price             10000 non-null  float64
 1   it_seller_rating  10000 non-null  float64
 2   it_quantity       10000 non-null  float64
dtypes: float64(3)
memory usage: 234.4 KB
```

[21]:
```
rmse(allegro_num.it_seller_rating, filled_data.it_seller_rating, squared=False)
```

[21]:
```
12773.57989845751
```

[22]:
```
vals = []

for i in range(10):
    indexes = allegro_num.sample(int(len(allegro_num) * 0.1)).index
    lost_data = allegro_num.copy(deep = True)
    lost_data.loc[indexes,'it_seller_rating'] = np.nan
    imputer = KNNImputer(n_neighbors = 3,weights = 'uniform').
 ↪fit_transform(lost_data)
    filled_data = pd.DataFrame(imputer)
    filled_data.columns = ['price', 'it_seller_rating', 'it_quantity']

    vals.append(rmse(allegro_num.it_seller_rating, filled_data.
 ↪it_seller_rating, squared=False))

stdev(vals)
```

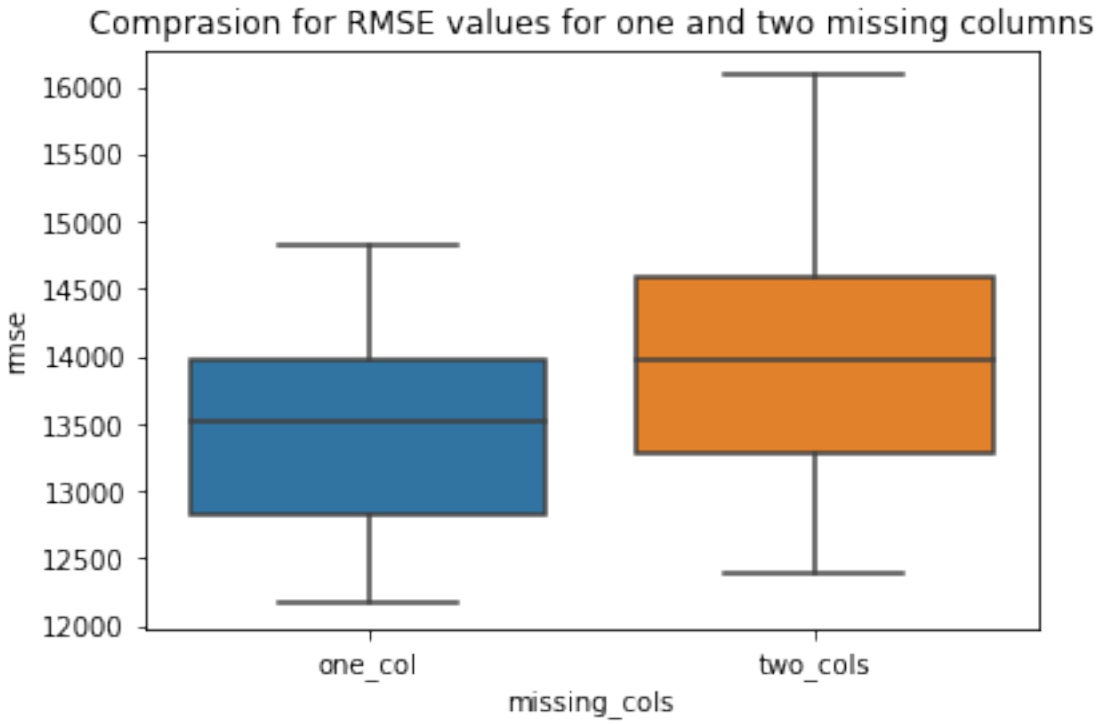
[22]:
```
827.5691278888557
```

[23]:
```
allegro_num = allegro_num.sample(5000)
#musiałem jeszcze bardziej ograniczyć dane, żeby użyć imputera dla dwóch kolumn

vals2 = []

for i in range(10):
    indexes = allegro_num.sample(int(len(allegro_num) * 0.1)).index
    lost_data = allegro_num.copy(deep = True)
    lost_data.loc[indexes,'it_seller_rating'] = np.nan
    indexes = allegro_num.sample(int(len(allegro_num) * 0.1)).index
    lost_data.loc[indexes,'it_quantity'] = np.nan

    imputer = KNNImputer(n_neighbors = 3,weights = 'uniform').
 ↪fit_transform(lost_data)
    filled_data = pd.DataFrame(imputer)
    filled_data.columns = ['price', 'it_seller_rating', 'it_quantity']

    vals2.append(rmse(allegro_num.it_seller_rating, filled_data.
 ↪it_seller_rating, squared=False))

stdev(vals2)
```

[23]:
```
1052.7431517774564
```

[24]:
```
dict = {'missing_cols': np.concatenate([['one_col']*10, ['two_cols']*10]),
     'rmse': np.concatenate([vals, vals2])}
df = pd.DataFrame(data=dict)
```

```
fig, ax = plt.subplots(figsize=(6, 4))
sns.boxplot(data = df, x = 'missing_cols', y = 'rmse', ax = ax)
plt.title('Comprasion for RMSE values for one and two missing columns')
plt.show()
```



Wartości RMSE są bardzo duże, czyli imputacja działa słabo.

Uzupełnienie brakujących wartości w dwóch kolumnach daje gorsze wyniki niż w jednej.