Untitled3

May 11, 2021

[1]:
```
import dalex as dx
import numpy as np
import pandas as pd

from sklearn.datasets import load_boston
from sklearn.model_selection import train_test_split, RandomizedSearchCV
from sklearn.svm import SVR
from sklearn.metrics import r2_score, mean_squared_error
from sklearn.preprocessing import MinMaxScaler

import warnings
warnings.filterwarnings('ignore')
```

[2]:
```
dalex_df = dx.datasets.load_apartments()
dalex_df.head()
```

[2]:

	m2_price	construction_year	surface	floor	no_rooms	district
1	5897	1953	25	3	1	Srodmiescie
2	1818	1992	143	9	5	Bielany
3	3643	1937	56	1	2	Praga
4	3517	1995	93	7	3	Ochota
5	3013	1992	144	6	5	Mokotow

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

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1000 entries, 1 to 1000
Data columns (total 6 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   m2_price           1000 non-null   int64
 1   construction_year  1000 non-null   int64
 2   surface            1000 non-null   int64
 3   floor              1000 non-null   int64
 4   no_rooms           1000 non-null   int64
 5   district           1000 non-null   object
dtypes: int64(5), object(1)
memory usage: 54.7+ KB
```

Jako drugi zbiór danych wziąłem zbiór dotyczący mieszkań w Bostonie z Lab1.

[4]:
```
boston_dict = load_boston()
boston_df = pd.DataFrame(boston_dict['data'],␣
↪columns=boston_dict['feature_names'])
boston_df['MEDV'] = boston_dict['target']

boston_df.head()
```

[4]:
```
      CRIM    ZN  INDUS  CHAS    NOX     RM   AGE     DIS  RAD    TAX  \
0  0.00632  18.0   2.31   0.0  0.538  6.575  65.2  4.0900  1.0  296.0
1  0.02731   0.0   7.07   0.0  0.469  6.421  78.9  4.9671  2.0  242.0
2  0.02729   0.0   7.07   0.0  0.469  7.185  61.1  4.9671  2.0  242.0
3  0.03237   0.0   2.18   0.0  0.458  6.998  45.8  6.0622  3.0  222.0
4  0.06905   0.0   2.18   0.0  0.458  7.147  54.2  6.0622  3.0  222.0

   PTRATIO       B  LSTAT  MEDV
0     15.3  396.90   4.98  24.0
1     17.8  396.90   9.14  21.6
2     17.8  392.83   4.03  34.7
3     18.7  394.63   2.94  33.4
4     18.7  396.90   5.33  36.2
```

[5]:
```
boston_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 506 entries, 0 to 505
Data columns (total 14 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   CRIM     506 non-null    float64
 1   ZN       506 non-null    float64
 2   INDUS    506 non-null    float64
 3   CHAS     506 non-null    float64
 4   NOX      506 non-null    float64
 5   RM       506 non-null    float64
 6   AGE      506 non-null    float64
 7   DIS      506 non-null    float64
 8   RAD      506 non-null    float64
 9   TAX      506 non-null    float64
 10  PTRATIO  506 non-null    float64
 11  B        506 non-null    float64
 12  LSTAT    506 non-null    float64
 13  MEDV     506 non-null    float64
dtypes: float64(14)
memory usage: 55.5 KB
```

[6]:
```
len(dalex_df['district'].unique())
```

[6]: 10

Ponieważ jest tylko 10 kategorii w ramce DALEX, użyjemy one-hot encodingu.

[7]:
```
dalex_df_enc = pd.concat([
    pd.get_dummies(dalex_df.district, prefix='District'),
    dalex_df], axis=1).drop(['district'], axis=1)

# zmieńmy jeszcze kolejność kolumn na bardziej intuicyjną

cols = dalex_df_enc.columns.tolist()
cols = cols[-4:] + cols[:-4]

dalex_df_enc = dalex_df_enc[cols]
dalex_df_enc.head()
```

[7]:
```
   construction_year  surface  floor  no_rooms  District_Bemowo  \
1               1953       25      3         1                0
2               1992      143      9         5                0
3               1937       56      1         2                0
4               1995       93      7         3                0
5               1992      144      6         5                0

   District_Bielany  District_Mokotow  District_Ochota  District_Praga  \
1                 0                 0                0               0
2                 1                 0                0               0
3                 0                 0                0               1
4                 0                 0                1               0
5                 0                 1                0               0

   District_Srodmiescie  District_Ursus  District_Ursynow  District_Wola  \
1                     1               0                 0              0
2                     0               0                 0              0
3                     0               0                 0              0
4                     0               0                 0              0
5                     0               0                 0              0

   District_Zoliborz  m2_price
1                  0      5897
2                  0      1818
3                  0      3643
4                  0      3517
5                  0      3013
```

[8]:
```
X_dalex = dalex_df_enc.drop('m2_price', axis=1)
Y_dalex = dalex_df_enc.m2_price

X_boston = boston_df.drop(['MEDV'], axis=1)
```

```
Y_boston = boston_df['MEDV']

X_train_dalex, X_test_dalex, y_train_dalex, y_test_dalex = train_test_split(
    X_dalex, Y_dalex, test_size = 0.33, random_state = 34)

X_train_boston, X_test_boston, y_train_boston, y_test_boston = train_test_split(
    X_boston, Y_boston, test_size = 0.33, random_state = 34)
```

1 SVM

[9]:
```
svm = SVR()
svm.fit(X_train_dalex, y_train_dalex)
y_hat_dalex = svm.predict(X_test_dalex)
print("Dalex")
print("Wynik R2: " + str(r2_score(y_test_dalex, y_hat_dalex)))
print("Miara RMSE: " + str(mean_squared_error(y_test_dalex, y_hat_dalex,␣
 ↪squared = False)))
```

```
Dalex
Wynik R2: -0.0035647663450799616
Miara RMSE: 934.3010814278865
```

[10]:
```
# przeskalujmy nasze dane i ponownie zbudujmy model
scaler = MinMaxScaler()
dalex_df_enc[['construction_year', 'surface', 'floor', 'no_rooms']] = scaler.
 ↪fit_transform(dalex_df_enc[[
    'construction_year', 'surface', 'floor', 'no_rooms']])

X_dalex = dalex_df_enc.drop('m2_price', axis=1)
Y_dalex = dalex_df_enc.m2_price

X_train_dalex, X_test_dalex, y_train_dalex, y_test_dalex = train_test_split(
    X_dalex, Y_dalex, test_size = 0.33, random_state = 34)
```

[11]:
```
svm = SVR()
svm.fit(X_train_dalex, y_train_dalex)
y_hat_dalex = svm.predict(X_test_dalex)
print("Dalex po przeskalowaniu")
print("Wynik R2: " + str(r2_score(y_test_dalex, y_hat_dalex)))
print("Miara RMSE: " + str(mean_squared_error(y_test_dalex, y_hat_dalex,␣
 ↪squared = False)))
```

```
Dalex po przeskalowaniu
Wynik R2: 0.040746588319345856
Miara RMSE: 913.441681296815
```

Widzimy, że po przeskalowaniu wyniki modelu uległy poprawieniu. Ten sam eksperyment przeprowadźmy dla datasetu bostońskiego

[12]:
```
svm_boston = SVR()
svm_boston.fit(X_train_boston, y_train_boston)
y_hat_boston = svm_boston.predict(X_test_boston)
print("Boston")
print("Wynik R2: " + str(r2_score(y_test_boston, y_hat_boston)))
print("Miara RMSE: " + str(mean_squared_error(y_test_boston, y_hat_boston,␣
 ↪squared = False)))
```

```
Boston
Wynik R2: 0.25006369536003814
Miara RMSE: 7.915412693509835
```

[13]:
```
scaler = MinMaxScaler()
boston_df[['CRIM', 'ZN', 'INDUS', 'CHAS', 'NOX', 'RM', 'AGE', 'DIS', 'RAD',␣
 ↪'TAX', 'PTRATIO', 'B', 'LSTAT']] = scaler.fit_transform(boston_df[['CRIM',␣
 ↪'ZN', 'INDUS', 'CHAS', 'NOX', 'RM', 'AGE', 'DIS', 'RAD', 'TAX', 'PTRATIO',␣
 ↪'B', 'LSTAT']])

X_boston = boston_df.drop('MEDV', axis=1)
Y_boston = boston_df.MEDV

X_train_boston, X_test_boston, y_train_boston, y_test_boston = train_test_split(
    X_boston, Y_boston, test_size = 0.33, random_state = 34)

svm = SVR()
svm.fit(X_train_boston, y_train_boston)
y_hat_boston = svm.predict(X_test_boston)
print("Boston po przeskalowaniu")
print("Wynik R2: " + str(r2_score(y_test_boston, y_hat_boston)))
print("Miara RMSE: " + str(mean_squared_error(y_test_boston, y_hat_boston,␣
 ↪squared = False)))
```

```
Boston po przeskalowaniu
Wynik R2: 0.6089303140466609
Miara RMSE: 5.71595038091061
```

Wniosek: Skalowanie danych przynosi dobre efekty.

2 Random Search

[14]:
```
parameters = dict(
    C = np.arange(start = 0.1, stop = 10000, step = 0.05),
    gamma = ['scale', 'auto'],
    degree = np.arange(1, 80, 1))

svm_rand_dalex = RandomizedSearchCV(svm_boston, parameters, cv=3, n_iter=200)

svm_rand_dalex.fit(X_train_dalex, y_train_dalex)
```

```python
print("Najlepsze parametry: " + str(svm_rand_dalex.best_params_))

best_estimator = svm_rand_dalex.best_estimator_
print("Wynik R2: " + str(r2_score(y_test_dalex, best_estimator.
 ↪predict(X_test_dalex))))
print(f'RMSE: {mean_squared_error(y_test_dalex, best_estimator.
 ↪predict(X_test_dalex), squared=False)}')
```

```
Najlepsze parametry: {'gamma': 'scale', 'degree': 5, 'C': 4608.550000000002}
Wynik R2: 0.9708321322696781
RMSE: 159.28192280776173
```

[15]:

```python
svm_rand_boston = RandomizedSearchCV(svm_boston, parameters, cv=3, n_iter=200)

svm_rand_boston.fit(X_train_boston, y_train_boston)
print("Najlepsze parametry: " + str(svm_rand_boston.best_params_))

best_estimator = svm_rand_boston.best_estimator_
print("Wynik R2: " + str(r2_score(y_test_boston, best_estimator.
 ↪predict(X_test_boston))))
print(f'RMSE: {mean_squared_error(y_test_boston, best_estimator.
 ↪predict(X_test_boston), squared=False)}')
```

```
Najlepsze parametry: {'gamma': 'scale', 'degree': 72, 'C': 196.25000000000006}
Wynik R2: 0.9054813042861667
RMSE: 2.810090028250059
```

[ ]:

```python

```