# spytek_mikolaj_pd4

May 4, 2021

## 1 Mikołaj Spytek - praca domowa 4

[2]:
```
import dalex as dx
import pandas as pd
from scipy.stats import uniform
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score
from sklearn.model_selection import RandomizedSearchCV
from sklearn.preprocessing import StandardScaler
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split
import warnings
warnings.filterwarnings('ignore')
```

```
Collecting dalex
  Downloading https://files.pythonhosted.org/packages/f3/50/5ad59eccfe1d4f
d86a9518929ad0c16c7ddb5575cdbf93fcc539e77e177c/dalex-1.1.0.tar.gz (966kB)
     |                  | 972kB 5.3MB/s
Requirement already satisfied: setuptools in
/usr/local/lib/python3.7/dist-packages (from dalex) (56.0.0)
Requirement already satisfied: pandas>=1.1.2 in /usr/local/lib/python3.7/dist-
packages (from dalex) (1.1.5)
Requirement already satisfied: numpy>=1.18.4 in /usr/local/lib/python3.7/dist-
packages (from dalex) (1.19.5)
Collecting plotly>=4.12.0
  Downloading https://files.pythonhosted.org/packages/1f/f6/bd3c17c8003b66
41df1228e80e1acac97ed8402635e46c2571f8e1ef63af/plotly-4.14.3-py2.py3-none-
any.whl (13.2MB)
     |                  | 13.2MB 270kB/s
Collecting tqdm>=4.48.2
  Downloading https://files.pythonhosted.org/packages/72/8a/34efae5cf99243
28a8f34eeb2fdaae14c011462d9f0e3fcded48e1266d1c/tqdm-4.60.0-py2.py3-none-any.whl
(75kB)
     |                  | 81kB 7.1MB/s
Requirement already satisfied: pytz>=2017.2 in
/usr/local/lib/python3.7/dist-packages (from pandas>=1.1.2->dalex) (2018.9)
Requirement already satisfied: python-dateutil>=2.7.3 in
/usr/local/lib/python3.7/dist-packages (from pandas>=1.1.2->dalex) (2.8.1)
```

```
Requirement already satisfied: six in /usr/local/lib/python3.7/dist-packages
(from plotly>=4.12.0->dalex) (1.15.0)
Requirement already satisfied: retrying>=1.3.3 in /usr/local/lib/python3.7/dist-
packages (from plotly>=4.12.0->dalex) (1.3.3)
Building wheels for collected packages: dalex
  Building wheel for dalex (setup.py) … done
  Created wheel for dalex: filename=dalex-1.1.0-cp37-none-any.whl size=982048
sha256=b0239675854f2600a9fe7dedd04f345e8bbfea87d292e20ff8758a727d5aeda6
  Stored in directory: /root/.cache/pip/wheels/3a/2d/5d/af76301c5b7e4c5916eb7b7a
bca0e2f62519584f325b80b370
Successfully built dalex
Installing collected packages: plotly, tqdm, dalex
  Found existing installation: plotly 4.4.1
    Uninstalling plotly-4.4.1:
      Successfully uninstalled plotly-4.4.1
  Found existing installation: tqdm 4.41.1
    Uninstalling tqdm-4.41.1:
      Successfully uninstalled tqdm-4.41.1
Successfully installed dalex-1.1.0 plotly-4.14.3 tqdm-4.60.0
```

[3]:
```python
#załadowanie zbiorów danych
apartments = dx.datasets.load_apartments()
apartments_test = dx.datasets.load_apartments_test()

x_apartments_train = apartments.iloc[:,0:4]
y_apartments_train = apartments.iloc[:, 5]
x_apartments_test = apartments_test.iloc[:,0:4]
y_apartments_test = apartments_test.iloc[:, 5]


data = load_breast_cancer()

x_cancer = pd.DataFrame(data=data.data, columns=data.feature_names)
y_cancer = pd.DataFrame(data.target)

x_cancer_train,  x_cancer_test, y_cancer_train, y_cancer_test =␣
 ↪train_test_split(x_cancer, y_cancer, random_state=42)
```

[4]:
```python
#przeskalowanie danych
apartments_scaler = StandardScaler()
x_apartments_train_scaled =  apartments_scaler.fit_transform(x_apartments_train)
x_apartments_test_scaled =  apartments_scaler.fit_transform(x_apartments_test)

cancer_scaler = StandardScaler()
x_cancer_train_scaled = cancer_scaler.fit_transform(x_cancer_train)
x_cancer_test_scaled = cancer_scaler.fit_transform(x_cancer_test)
```

[5]:
```
#wytrenowanie modeli na danych z i bez skalowania
sv_a1 = SVC(kernel="rbf", random_state=42)
sv_a2 = SVC(kernel="rbf", random_state=42)
sv_c1 = SVC(kernel="rbf", random_state=42)
sv_c2 = SVC(kernel="rbf", random_state=42)

sv_a1.fit(x_apartments_train, y_apartments_train)
unscaled_apartments_pred = sv_a1.predict(x_apartments_test)

sv_a2.fit(x_apartments_train_scaled, y_apartments_train)
scaled_apartments_pred = sv_a1.predict(x_apartments_test_scaled)


sv_c1.fit(x_cancer_train, y_cancer_train)
unscaled_cancer_pred = sv_c1.predict(x_cancer_test)

sv_c2.fit(x_cancer_train_scaled, y_cancer_train)
scaled_cancer_pred = sv_c1.predict(x_cancer_test_scaled)

unscaled_apartments_acc = accuracy_score(y_apartments_test,␣
 ↪unscaled_apartments_pred)
scaled_apartments_acc = accuracy_score(y_apartments_test,␣
 ↪scaled_apartments_pred)
unscaled_cancer_acc = accuracy_score(y_cancer_test, unscaled_cancer_pred)
scaled_cancer_acc = accuracy_score(y_cancer_test, scaled_cancer_pred)

results = [[unscaled_apartments_acc,scaled_apartments_acc],
           [unscaled_cancer_acc,scaled_cancer_acc]]


pd_results =pd.DataFrame(data=results, columns=["unscaled","scaled"],␣
 ↪index=["apartments","cancer"])

pd_results
```

[5]:
```
            unscaled    scaled
apartments  0.220778  0.099111
cancer      0.951049  0.622378
```

Jak widać skalowanie nie dało pozytywnego efektu. Zbiór danych dotyczący raka piersi był już przeskalowany. Dodatkowo wg. dokumentacji sklearna SVM również przeprowadza skalowanie. Takie potrójne skalowanie przynosi więc ujemne efekty. Jeśli sprawa tyczy się zbioru apartments skalowanie również nie przynosi pożądanych efektów.

[6]:
```
#hiperparametry do optymalizacji
distributions = dict(C= uniform(loc=0, scale=4),
                     degree=[i for i in range(1,15)],
                        gamma = uniform(loc=0, scale=1)
                       )
```

[7]:
```
clf_a = RandomizedSearchCV(sv_a1, distributions, n_iter=1000, verbose=True,␣
 ↪random_state=42)
search_a = clf_a.fit(x_apartments_train, y_apartments_train)

clf_c = RandomizedSearchCV(sv_c1, distributions, n_iter=1000, verbose=True,␣
 ↪random_state=42)
search_c = clf_c.fit(x_cancer_train, y_cancer_train)
```

```
Fitting 5 folds for each of 1000 candidates, totalling 5000 fits

[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done 5000 out of 5000 | elapsed:  5.2min finished

Fitting 5 folds for each of 1000 candidates, totalling 5000 fits

[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done 5000 out of 5000 | elapsed:  1.2min finished
```

[8]:
```
search_a.best_params_
```

[8]:
```
{'C': 3.494314496427109, 'degree': 13, 'gamma': 0.0025950243836465603}
```

[9]:
```
search_c.best_params_
```

[9]:
```
{'C': 1.919588326368293, 'degree': 11, 'gamma': 0.004939980934409616}
```

[10]:
```
#sprawdzenie, czy te parametry poprawiają model
sv_a_new = SVC(kernel="rbf",C=3.5589922261307083, degree=6, gamma=0.
 ↪0013385008146062916, random_state=42)
sv_a_new.fit(x_apartments_train, y_apartments_train)
unscaled_apartments_pred_new = sv_a_new.predict(x_apartments_test)
accuracy_score(y_apartments_test, unscaled_apartments_pred_new)
```

[10]:
```
0.23766666666666666
```

Jak widać, w tym przypadku mamy niewielki zysk.

[11]:
```
sv_c_new = SVC(kernel="rbf", C= 1.8800995309030366, degree= 5, gamma=0.
 ↪0010431294303261396, random_state=42)
sv_c_new.fit(x_cancer_train, y_cancer_train)
unscaled_cancer_pred_new = sv_c_new.predict(x_cancer_test)
accuracy_score(y_cancer_test, unscaled_cancer_pred_new)
```

[11]:
```
0.9230769230769231
```

W tym przypadku, zmiana hiperparametrów pogorszyła wynik. Być może dlatego, że już z domyślnymi parametrami model osiągnął tak dobry wynik, albo random search działał zbyt krótko i nie wystarczająco przeszukał przestrzeń parametrów.