PD4

May 5, 2021

[1]:
```
import pandas as pd
from sklearn.svm import SVC
from sklearn.datasets import load_wine
import dalex
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.model_selection import RandomizedSearchCV
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import accuracy_score
import warnings
warnings.filterwarnings('ignore')
```

1 Zbiór danych apartments

[2]:
```
aparts = dalex.datasets.load_apartments()
print(aparts.info())
aparts.head()
```

```
<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
None
```

[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

1.0.1 Encoding i podział na zbiory treningowy i testowy

[23]:
```
le=LabelEncoder()
aparts['district'] = le.fit_transform(aparts['district'])

X = aparts.drop('district', axis = 1)
y = aparts['district']

X_train, X_test, y_train, y_test = train_test_split(X, y, stratify=y, test_size␣
 ↪= 0.3, random_state = 29)
```

1.0.2 Prosty model bez skalowania

[15]:
```
model_svc = SVC()

model_svc.fit(X_train, y_train)
y_pred = model_svc.predict(X_test)

simple_model = accuracy_score(y_test, y_pred)
print('Accuracy score dla prostego modelu bez skalowania i strojenia␣
 ↪hiperparametrów wynosi: ',  simple_model)
```

```
Accuracy score dla prostego modelu bez skalowania i strojenia hiperparametrów
wynosi:  0.21666666666666667
```

1.0.3 Strojenie hiperparametrów bez skalowania

[24]:
```
model_svc = SVC()

C = [i for i in range(10)]
C.append(20)
C.append(50)

parameters = {
    'C' : C,
    'gamma' : ['scale', 'auto', 3, 10],
    'degree': np.linspace(1, 5, 5)
```

```
}

svc_rand = RandomizedSearchCV(model_svc, parameters, n_iter=1000, n_jobs=-1)
best_svc = svc_rand.fit(X_train ,y_train)
best_svc.best_estimator_
```

[24]:
```
SVC(C=1, degree=1.0)
```

[25]:
```
model_svc = best_svc.best_estimator_
model_svc.fit(X_train, y_train)

y_pred = model_svc.predict(X_test)

hiper_model = accuracy_score(y_test, y_pred)
print('Accuracy score dla modelu bez skalowania ze strojeniem hiperparametrów␣
 ↪wynosi: ',  hiper_model)
```

```
Accuracy score dla modelu bez skalowania ze strojeniem hiperparametrów wynosi:
0.21666666666666667
```

1.0.4 Skalowanie bez strojenia hiperparametrów

[27]:
```
scaler = StandardScaler()
X_train =  scaler.fit_transform(X_train)
X_test =  scaler.fit_transform(X_test)

model_svc = SVC()

model_svc.fit(X_train, y_train)
y_pred = model_svc.predict(X_test)

scal_model = accuracy_score(y_test, y_pred)
print('Accuracy score dla prostego modelu ze skalowaniem i bez strojenia␣
 ↪hiperparametrów wynosi: ',  scal_model)
```

```
Accuracy score dla prostego modelu ze skalowaniem i bez strojenia
hiperparametrów wynosi:  0.31333333333333335
```

1.0.5 Strojenie hiperparametrów ze skalowaniem

[30]:
```
model_svc = SVC()

svc_rand = RandomizedSearchCV(model_svc, parameters, n_iter=1000, n_jobs=-1)
best_svm = svc_rand.fit(X_train ,y_train)
best_svm.best_estimator_
```

[30]:
```
SVC(C=4, degree=1.0, gamma='auto')
```

[31]:
```
model_svc = best_svm.best_estimator_

model_svc.fit(X_train, y_train)

y_pred = model_svc.predict(X_test)

hiper_scal_model = mean_squared_error(y_test, y_pred, squared=True)
print('RMSE dla modelu ze skalowaniem i strojeniem hiperparametrów wynosi: ', ␣
 ↪hiper_scal_model)
```

```
RMSE dla modelu ze skalowaniem i strojeniem hiperparametrów wynosi:
16.546666666666667
```

1.0.6 Podsumowanie

Skalowanie poprawiło wyniki modelu, za to strojenie hiperparametrów bez skalowania nie poprawiło wyniku, a po skalowaniu nawet pogorszyło, może to wynikać z tego, że RandomizedSearch nie trafił w optymalne rozwiązanie.

2 Zbiór danych Wine

[13]:
```
wine_dict = load_wine()
```

[14]:
```
X = pd.DataFrame(wine_dict['data'], columns = wine_dict['feature_names'])
Y = wine_dict['target']

X_train, X_test, y_train, y_test = train_test_split(X, Y, random_state = 29,␣
 ↪test_size = 0.3)
```

2.0.1 Prosty model bez skalowania

[15]:
```
model_svc = SVC(random_state=29)

model_svc.fit(X_train, y_train)
y_pred = model_svc.predict(X_test)

simple_model = accuracy_score(y_test, y_pred)

print('Accuracy score dla prostego modelu bez skalowania i strojenia␣
 ↪hiperparametrów wynosi: ',  simple_model)
```

```
Accuracy score dla prostego modelu bez skalowania i strojenia hiperparametrów
wynosi:  0.5740740740740741
```

2.0.2 Strojenie hiperparametrów bez skalowania

[16]:
```
model_svc = SVC()

parameters = {
    'C' : [np.linspace(1, 10, 10), 20, 50],
    'gamma' : ['scale', 'auto', 3, 10],
    'degree': np.linspace(1, 5, 5),
    'kernel': ['linear', 'poly', 'rbf']
}

svc_rand = RandomizedSearchCV(model_svc, parameters,  n_iter=100)
best_svc = svc_rand.fit(X_train ,y_train)
best_svc.best_estimator_
```

[16]:
```
SVC(C=20, degree=4.0, gamma=3, kernel='poly')
```

[17]:
```
model_svc = best_svc.best_estimator_

model_svc.fit(X_train, y_train)

y_pred = model_svc.predict(X_test)

hiper_model = accuracy_score(y_test, y_pred)
print('Accuracy score dla modelu bez skalowania ze strojeniem hiperparametrów␣
 ↪wynosi: ',  hiper_model)
```

```
Accuracy score dla modelu bez skalowania ze strojeniem hiperparametrów wynosi:
0.9259259259259259
```

2.0.3 Skalowanie bez strojenia hiperparametrów

[18]:
```
scaler = StandardScaler()
X_train =  scaler.fit_transform(X_train)
X_test =  scaler.fit_transform(X_test)

model_svc = SVC(random_state=29)

model_svc.fit(X_train, y_train)
y_pred = model_svc.predict(X_test)

scal_model = accuracy_score(y_test, y_pred)
```

```
print('Accuracy score dla prostego modelu ze skalowaniem i bez strojenia␣
 ↪hiperparametrów wynosi: ',  scal_model)
```

```
Accuracy score dla prostego modelu ze skalowaniem i bez strojenia
hiperparametrów wynosi:  0.9814814814814815
```

2.0.4 Strojenie hiperparametrów ze skalowaniem

[19]:
```
model_svc = SVC()
svc_rand = RandomizedSearchCV(model_svc, parameters,  n_iter=100,␣
 ↪random_state=29)
best_svc = svc_rand.fit(X_train ,y_train)
best_svc.best_estimator_
```

[19]:
```
SVC(C=20, degree=4.0, gamma='auto', kernel='linear')
```

[20]:
```
model_svc = best_svc.best_estimator_

model_svc.fit(X_train, y_train)

y_pred = model_svc.predict(X_test)

hiper_model = accuracy_score(y_test, y_pred)
print('Accuracy score dla modelu ze skalowaniem i strojeniem hiperparametrów␣
 ↪wynosi: ',  hiper_model)
```

```
Accuracy score dla modelu ze skalowaniem i strojeniem hiperparametrów wynosi:
0.9259259259259259
```

2.0.5 Podsumowanie

Skalowanie sprawiło, że model ma prawie 98% poprawności, jednak stojenie hiperparametrów nie poprawiło wyników.