Praca_domowa_3

April 10, 2021

1 Praca domowa 3

Bartosz Sawicki

[1]:
```
import pandas as pd
```

1.1 Wczytanie zbioru danych

[2]:
```
path = '../../australia.csv'

df = pd.read_csv(path)
```

[3]:
```
df.head()
```

[3]:

	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustSpeed	WindSpeed9am	WindSpeed3pm	Humidity9am	Humidity3pm	Pressure9am	Pressure3pm	Cloud9am	Cloud3pm	Temp9am	Temp3pm	RainToday	RainTomorrow
0	17.9	35.2	0.0	12.0	12.3	48.0	6.0	20.0	20.0	13.0	1006.3	1004.4	2.0	5.0	26.6	33.4	0	0
1	18.4	28.9	0.0	14.8	13.0	37.0	19.0	19.0	30.0	8.0	1012.9	1012.1	1.0	1.0	20.3	27.0	0	0
2	19.4	37.6	0.0	10.8	10.6	46.0	30.0	15.0	42.0	22.0	1012.3	1009.2	1.0	6.0	28.7	34.9	0	0
3	21.9	38.4	0.0	11.4	12.2	31.0	6.0	6.0	37.0	22.0	1012.7	1009.1	1.0	5.0	29.1	35.6	0	0
4	24.2	41.0	0.0	11.2	8.4	35.0	17.0	13.0	19.0	15.0	1010.7	1007.4	1.0	6.0	33.6	37.6	0	0

[4]:
```
df.shape
```

[4]:
```
(56420, 18)
```

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 56420 entries, 0 to 56419
Data columns (total 18 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   MinTemp        56420 non-null  float64
 1   MaxTemp        56420 non-null  float64
 2   Rainfall       56420 non-null  float64
 3   Evaporation    56420 non-null  float64
 4   Sunshine       56420 non-null  float64
 5   WindGustSpeed  56420 non-null  float64
 6   WindSpeed9am   56420 non-null  float64
 7   WindSpeed3pm   56420 non-null  float64
 8   Humidity9am    56420 non-null  float64
 9   Humidity3pm    56420 non-null  float64
 10  Pressure9am    56420 non-null  float64
 11  Pressure3pm    56420 non-null  float64
 12  Cloud9am       56420 non-null  float64
 13  Cloud3pm       56420 non-null  float64
 14  Temp9am        56420 non-null  float64
 15  Temp3pm        56420 non-null  float64
 16  RainToday      56420 non-null  int64  
 17  RainTomorrow   56420 non-null  int64  
dtypes: float64(16), int64(2)
memory usage: 7.7 MB
```

1.2 Podział na zbiór testowy i treningowy

[6]:
```
from sklearn.model_selection import train_test_split
```

[7]:
```
X = df.drop('RainTomorrow', axis='columns')
y = df.loc[:,'RainTomorrow']
display(X,y)
```

```
       MinTemp  MaxTemp  Rainfall  Evaporation  Sunshine  WindGustSpeed  \
0         17.9     35.2       0.0         12.0      12.3           48.0   
1         18.4     28.9       0.0         14.8      13.0           37.0   
2         19.4     37.6       0.0         10.8      10.6           46.0   
3         21.9     38.4       0.0         11.4      12.2           31.0   
4         24.2     41.0       0.0         11.2       8.4           35.0   
…          …        …         …           …         …              …     
56415     19.3     33.4       0.0          6.0      11.0           35.0   
56416     21.2     32.6       0.0          7.6       8.6           37.0   
56417     20.7     32.8       0.0          5.6      11.0           33.0   
56418     19.5     31.8       0.0          6.2      10.6           26.0   
```

```
56419     20.2     31.7       0.0          5.6      10.7          30.0

       WindSpeed9am  WindSpeed3pm  Humidity9am  Humidity3pm  Pressure9am  \
0               6.0          20.0         20.0         13.0       1006.3
1              19.0          19.0         30.0          8.0       1012.9
2              30.0          15.0         42.0         22.0       1012.3
3               6.0           6.0         37.0         22.0       1012.7
4              17.0          13.0         19.0         15.0       1010.7
…               …             …            …            …           …
56415           9.0          20.0         63.0         32.0       1013.9
56416          13.0          11.0         56.0         28.0       1014.6
56417          17.0          11.0         46.0         23.0       1015.3
56418           9.0          17.0         62.0         58.0       1014.9
56419          15.0           7.0         73.0         32.0       1013.9

       Pressure3pm  Cloud9am  Cloud3pm  Temp9am  Temp3pm  RainToday
0           1004.4       2.0       5.0     26.6     33.4          0
1           1012.1       1.0       1.0     20.3     27.0          0
2           1009.2       1.0       6.0     28.7     34.9          0
3           1009.1       1.0       5.0     29.1     35.6          0
4           1007.4       1.0       6.0     33.6     37.6          0
…            …       …        …       …        …        …
56415       1010.5       0.0       1.0     24.5     32.3          0
56416       1011.2       7.0       0.0     24.8     32.0          0
56417       1011.8       0.0       0.0     24.8     32.1          0
56418       1010.7       1.0       1.0     24.8     29.2          0
56419       1009.7       6.0       5.0     25.4     31.0          0

[56420 rows x 17 columns]

0        0
1        0
2        0
3        0
4        0
        ..
56415    0
56416    0
56417    0
56418    0
56419    0
Name: RainTomorrow, Length: 56420, dtype: int64
```

[8]:
```
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=1337,
    stratify=y) # dzielimy tak, żeby zbiór treningowy i testowy miały taki sam
 ↪rozkład y
```

1.3 Nauczenie klasyfikatorów

1.3.1 Regresja logistyczna z l2

[9]:
```
from sklearn.linear_model import LogisticRegression
```

[10]:
```
lr = LogisticRegression(penalty='l2', random_state=1337).fit(X_train, y_train)
```

```
/home/sawcio/Studia/4sem/Wstęp_do_U_M/venv/lib/python3.8/site-
packages/sklearn/linear_model/_logistic.py:763: ConvergenceWarning: lbfgs failed
to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
```

Przy domyślnym `max_iter`=100 regresja nie jest zbieżna.

[11]:
```
lr = LogisticRegression(penalty='l2', random_state=1337, max_iter=1000).
↪fit(X_train, y_train)
```

Po zwiększeniu parametru mamy zbieżność.

1.3.2 AdaBoost

[12]:
```
from sklearn.ensemble import AdaBoostClassifier
```

[13]:
```
ab = AdaBoostClassifier(n_estimators=200, random_state=1337).fit(X_train,␣
↪y_train)
```

1.3.3 KNN Clasifier

[14]:
```
from sklearn.neighbors import KNeighborsClassifier
```

[15]:
```
knn = KNeighborsClassifier(n_neighbors=7, p=1).fit(X_train, y_train)
```

1.4 Ewaluacja modeli

Używane metryki: - accuracy - balanced_accuracy - f1 - roc_auc

[16]:
```
from sklearn.metrics import accuracy_score, balanced_accuracy_score, f1_score,␣
↪roc_auc_score
```

[17]:
```
models = [lr,ab,knn]
metrics = [accuracy_score, balanced_accuracy_score, f1_score, roc_auc_score]
```

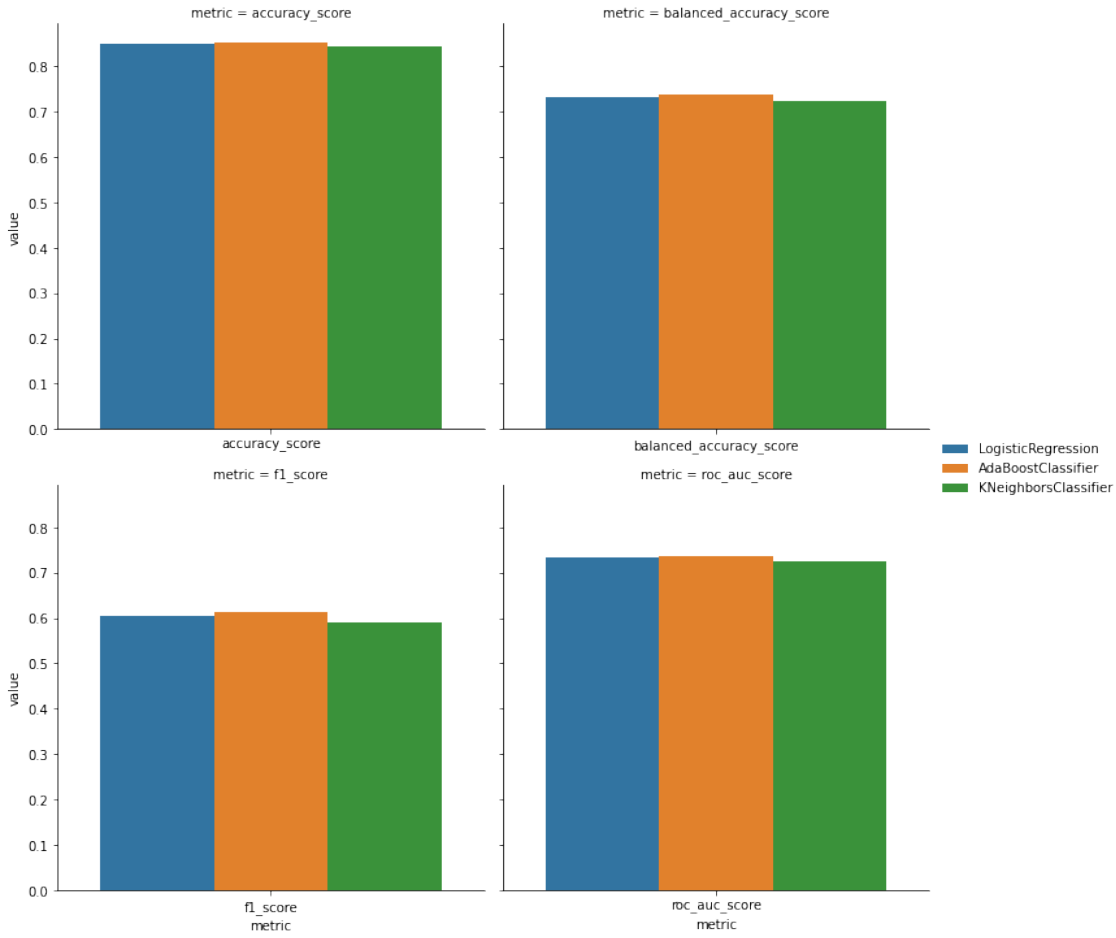
```
result_dict = {metric.__name__: {} for metric in metrics}

for model in models:
    y_pred = model.predict(X_test)
    for metric in metrics:
        result_dict.get(metric.__name__).update({type(model).__name__ :␣
 ↪metric(y_test, y_pred)})
```

[18]:
```
metric_names = [metric.__name__ for metric in metrics]
res = pd.DataFrame(result_dict)
res = pd.concat((res,res.index.to_series()), axis=1)
results = res.melt(0,metric_names)
results['metric'] = results['variable']
```

[19]:
```
import matplotlib.pyplot as plt
import seaborn as sns
```

[20]:
```
sns.catplot(data=results, hue=0, y='value', x='metric', kind='bar',␣
 ↪col='metric', sharex=False, col_wrap=2)
plt.show()
```



Niezależnie, czy patrzymy na wynik jednej metryki, czy na wszystkie, najlepszym klasyfikatorem okazuje się AdaBoost. Niemniej jego przewaga nad innymi modelami jest niewielka.