PD3

April 13, 2021

1 Zadanie Domowe 3 - Katarzyna Solawa

Zbiór danych zawiera codzienne obserwacje pogody z wielu australijskich stacji pogodowych. Waszym zadaniem będzie stworzenie modelu klasyfikacyjnego, który będzie potrafił przewidzieć czy następnego dnia będzie padał deszcz czy też nie.

[4]:
```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
import sklearn
from sklearn.metrics import f1_score
from sklearn.metrics import accuracy_score
from sklearn.metrics import precision_score
```

1.1 Wczytanie danych

[5]:
```
df = pd.read_csv("australia.csv")
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
```

[15]:
```
df.head()
```

[15]:

	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	

	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	

	Pressure3pm	Cloud9am	Cloud3pm	Temp9am	Temp3pm	RainToday	RainTomorrow
0	1004.4	2.0	5.0	26.6	33.4	0	0
1	1012.1	1.0	1.0	20.3	27.0	0	0
2	1009.2	1.0	6.0	28.7	34.9	0	0
3	1009.1	1.0	5.0	29.1	35.6	0	0
4	1007.4	1.0	6.0	33.6	37.6	0	0

1.2 Podział danych na zbiór treningowy i testowy

[6]:
```
y = np.array(df['RainTomorrow'])
X = df.drop(['RainTomorrow'],axis=1)
```

[7]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=42)
```

1.3 Nauczenie 3 dowolnych klasyfikatorów

1.4 W każdym klasyfikatorze należy wybrać minimum jeden hiperparametr

[8]:
```
from sklearn.svm import SVC
svm = SVC(kernel='sigmoid')

svm.fit(X_train,y_train)
y_hat_svm = svm.predict(X_test)
```

`kernel` - domyślnie 'rbf', możliwe do wyboru: 'linear', 'poly', 'rbf', 'sigmoid', 'precomputed'

[9]:
```
from sklearn.linear_model import LogisticRegression
lr = LogisticRegression(max_iter=5000, solver='sag')

lr.fit(X_train,y_train)
y_hat_logreg = lr.predict(X_test)
```

`solver` - Algorytm wykorzystywany w problemie optymalizacji , domyślnie równy 'lbfgs'

'sag' wspiera `penalty` = 'l2', co jest ustawieniem domyślnym

[10]:
```
from sklearn.tree import DecisionTreeClassifier,plot_tree #export_graphviz
## biblioteka poniżej może być problematyczna na Windows
#import graphviz

tree1 = DecisionTreeClassifier(splitter='random')

tree1.fit(X_train,y_train)
y_hat_tree = tree1.predict(X_test)
```

`splitter` -Strategia użyta do wyboru podziału w każdym węźle, domyślnie ma wartość 'best'.

`splitter`='random' wybiera najlepszy losowy podział

1.5 Wykorzystanie przynajmniej 3 miar oceny jakości klasyfikatorów i wybór najlepszego z nich

[11]:
```
print(f1_score(y_test,y_hat_svm))
print(accuracy_score(y_test, y_hat_svm))
print(precision_score(y_test, y_hat_svm))
```

```
0.0019175455417066156
0.7785891527827011
1.0
```

[12]:
```
print(f1_score(y_test,y_hat_logreg))
print(accuracy_score(y_test, y_hat_logreg))
print(precision_score(y_test, y_hat_logreg))
```

```
0.6168189446205482
0.8543069833392414
0.7393831023692445
```

[13]:
```
print(f1_score(y_test,y_hat_tree))
print(accuracy_score(y_test, y_hat_tree))
print(precision_score(y_test, y_hat_tree))
```

```
0.53577106518283
0.792981212336051
0.5325537294563844
```

Dla trzech klasyfiktaorów najwyższą miarę jakości otrzymaliśmy dla `Accuracy`. Najbardziej rozbierzne wyniki otrzymaliśmy dla klasyfikatora `SVM`

[14]:
```
from sklearn.metrics import confusion_matrix
conf_m = confusion_matrix(y_test, y_hat_svm)
pd.DataFrame(conf_m)
```

[14]:

	0	1
0	10979	0
1	3123	3

TN = 10979

FN = 3123

TP = 3

FP = 0

`Accuracy` mówi nam ile było poprawnych wyników w stosunku do całości, a `F1 score` głównie zwraca uwagę na TP, które w naszym przypadku jest małe. W naszych danych więszość wyników stanowi TN. Gdyby jako wynik pozytywny przyjąć brak deszczu następnego dnia, wtedy zarówno dla `Accuracy` oraz `F1 score`, otrzymamy wysoką miarę jakości. Podobnie dla `Precision` które sprawdza stosunek TP do sumy TP i FP.

[16]:
```
from sklearn.metrics import confusion_matrix
conf_m = confusion_matrix(y_test, y_hat_logreg)
pd.DataFrame(conf_m)
```

[16]:

	0	1
0	10396	583
1	1472	1654

[17]:
```
from sklearn.metrics import confusion_matrix
conf_m = confusion_matrix(y_test, y_hat_tree)
pd.DataFrame(conf_m)
```

[17]:

	0	1
0	9500	1479
1	1441	1685

`Accuracy` jezt zbyt ogólną miarą jakości, skupia się tylko na poprawnych wynikach. Jeżeli bardziej interesuje nas sprawdzenie jakości dla TP albo TN(wtedy trzeba zaminić wyniki negatywne z pozytywnymi), wiecej dowiemy się używając w odpowiedni sposób `F1 score` lub `Precision`

W zadaniu przewidzenia czy bedzię padać nastęnego dnia, wydaje mi się, że predykcja wystąpienia deszczu jest ważniejsza. Dlatego uważam, że najlepszą miarą bedzie `F1-score`

[ ]: