PrzemyslawOlenderPD3

April 13, 2021

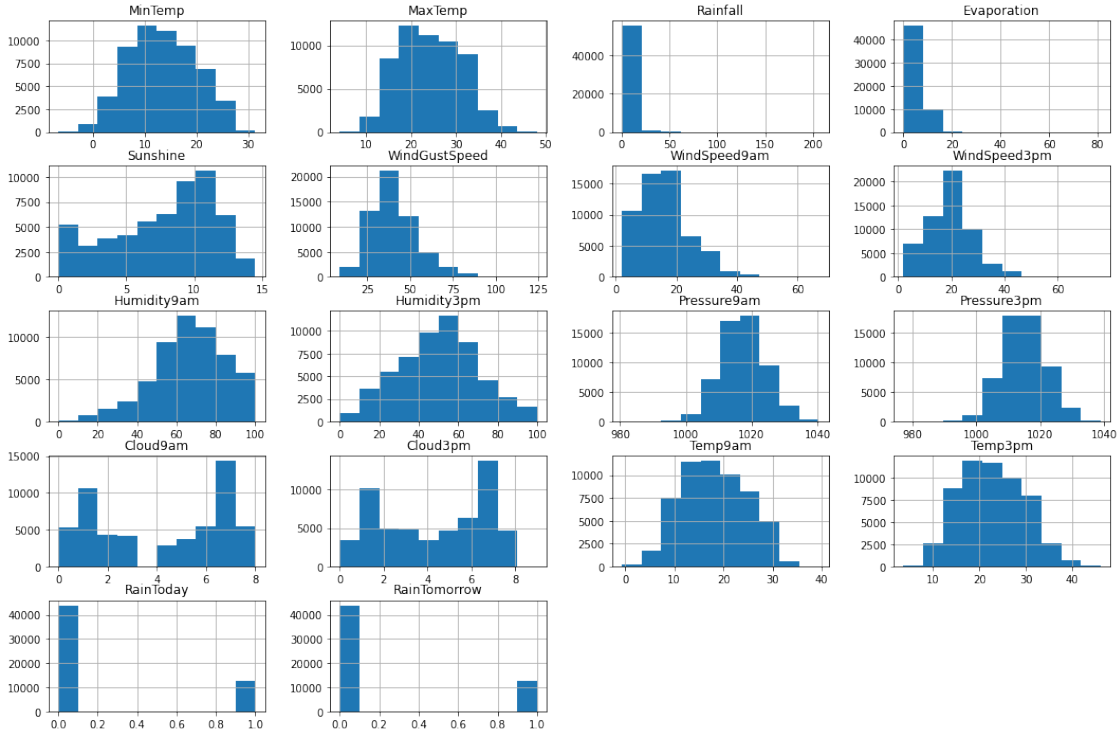
[1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import KFold, cross_validate
```

[2]:
```
rain_df = pd.read_csv("https://raw.githubusercontent.com/mini-pw/2021L-WUM/main/
↪Prace_domowe/Praca_domowa3/australia.csv")
rain_df.head()
```

[2]:

	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

[3]:
```
rain_df.hist(figsize = (18, 12))
plt.show()
```



Widać dużą dysproporcje miedzy dnami deszczowymi i bezdeszczowymi, uwzględnimy ją przy dzieleniu dancch na zbiory.

0.0.1 Podział na zbiór treningowy i testowy

[4]:
```
X = rain_df.drop('RainTomorrow', axis = 1)
y = rain_df[['RainTomorrow']]
```

[5]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, stratify = y,␣
 ↪test_size = 0.2, random_state = 29)

print(f'X_train shape: {X_train.shape}')
print(f'X_test shape:  {X_test.shape}')
print(f'y_train shape: {y_train.shape}')
print(f'y_test shape:  {y_test.shape}')
```

```
X_train shape: (45136, 17)
X_test shape:  (11284, 17)
y_train shape: (45136, 1)
y_test shape:  (11284, 1)
```

0.1 Uczenie modeli

[6]:
```
split = KFold(n_splits = 10, shuffle =True, random_state = 29)

def results_to_df(results, model):
    data = [
        ['accuracy', np.mean(results['test_accuracy'])],
        ['precision', np.mean(results['test_precision'])],
        ['roc_aux', np.mean(results['test_roc_auc'])],
        ['recall', np.mean(results['test_recall'])]
    ]
    return pd.DataFrame(data, columns = ['measure', model])
```

0.1.1 Regresja liniowa

[7]:
```
from sklearn.linear_model import LogisticRegression

lr = LogisticRegression(penalty = 'l1', solver='saga', random_state = 29,␣
↪max_iter = 1000)

lr_pipe = Pipeline(
        steps=[
        ('standardscaler', StandardScaler()),
        ('lr', lr)])

lr_results = cross_validate(lr_pipe, X_train, y_train, cv = split, n_jobs = -1,
                            scoring = ['accuracy', 'precision', 'roc_auc',␣
↪'recall'])


lr_df = results_to_df(lr_results, 'LogisticRegression')
```

[8]:
```
lr_df
```

[8]:

	measure	LogisticRegression
0	accuracy	0.853443
1	precision	0.728115
2	roc_aux	0.884563
3	recall	0.534535

0.1.2 Random Forest

[9]:
```
from sklearn.ensemble import RandomForestClassifier

randomForest = RandomForestClassifier(max_depth = 10, random_state = 29,␣
↪n_estimators = 300)

randomForest_pipe = Pipeline(
    steps=[
        ('standardscaler', StandardScaler()),
        ('rf', randomForest)])

randomForest_results = cross_validate(randomForest_pipe, X_train, y_train, cv =␣
↪split, n_jobs = -1,
                                scoring = ['accuracy', 'precision', 'roc_auc',␣
↪'recall'])


randomForest_df = results_to_df(randomForest_results, 'RandomForest')
```

[10]:
```
randomForest_df
```

[10]:

	measure	RandomForest
0	accuracy	0.855570
1	precision	0.763147
2	roc_aux	0.887460
3	recall	0.499548

0.1.3 K Nearest Neighbors

[11]:
```
from sklearn.neighbors import KNeighborsClassifier

knn = KNeighborsClassifier(n_neighbors = 10, algorithm = 'ball_tree', leaf_size␣
↪= 40)

knn_pipe = Pipeline(
    steps=[
        ('standardscaler', StandardScaler()),
        ('knn', knn)])

knn_results = cross_validate(knn_pipe, X_train, y_train, cv = split, n_jobs =␣
↪-1,
                                scoring = ['accuracy', 'precision', 'roc_auc',␣
↪'recall'])
```

```
knn_df = results_to_df(knn_results, 'KNeighbours')
```

[12]:
```
knn_df
```

[12]:

	measure	KNeighbours
0	accuracy	0.847505
1	precision	0.750117
2	roc_aux	0.859310
3	recall	0.461281

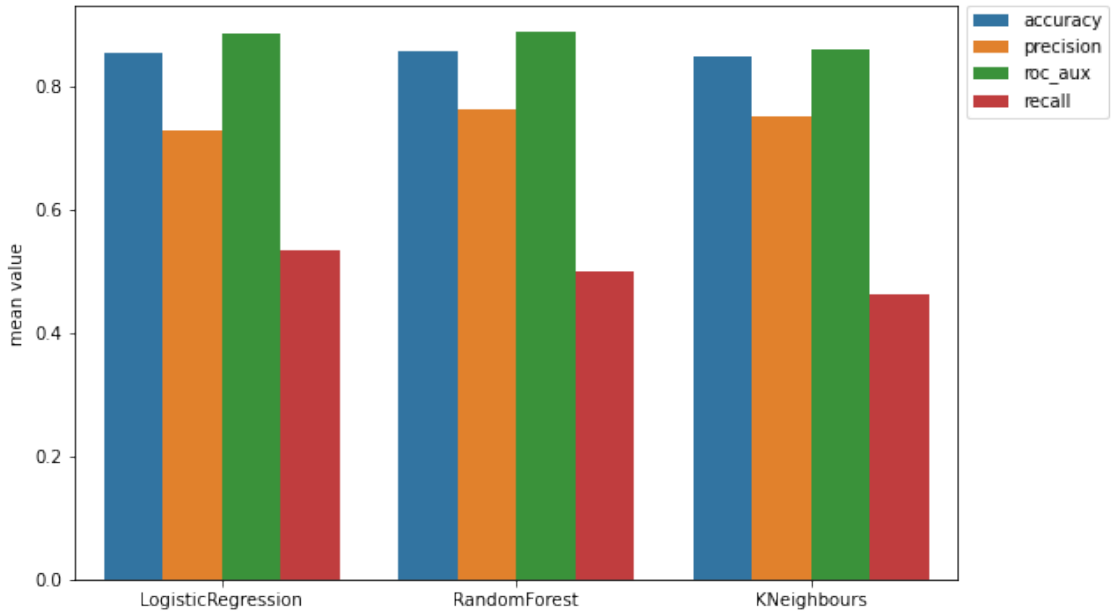
0.2 Ocena modeli

[13]:
```
all_results = pd.merge((pd.merge(lr_df, randomForest_df, on = 'measure')),␣
↪knn_df, on = 'measure')

all_results
```

[13]:

	measure	LogisticRegression	RandomForest	KNeighbours
0	accuracy	0.853443	0.855570	0.847505
1	precision	0.728115	0.763147	0.750117
2	roc_aux	0.884563	0.887460	0.859310
3	recall	0.534535	0.499548	0.461281

[14]:
```
all_results_melt = all_results.melt(id_vars=['measure'],
                                    value_vars=['LogisticRegression',␣
↪'RandomForest', 'KNeighbours'])

plt.figure(figsize = (9, 6))
sns.barplot(data = all_results_melt, x = 'variable', y = 'value', hue =␣
↪'measure'
           ).set(xlabel='', ylabel='mean value')
plt.legend(bbox_to_anchor=(1.01, 1),borderaxespad=0)
plt.show()
```



Dla każdego z modeli średnie wartośi mair są na podobnym poziomie, więc ciężko wybrać najlepszy s pośród nich.

Accuracy we wszystkich moedlach jest bardzo wysokie, jednak sprawdza ono tylko poprawność odpowiedzi, która należy do zbioru {0, 1}, przy czym 0 jest znacznie więcej niż 1. Nietrudo więc trafić w większość poprawnych odpowiedzi. Auc_roc daje równie wysokie wyniki.

Porównując Precision i Recall można zauważyć dysproporcje, dni bez deszczu są częściej są uznawane na deszczowe niż dni deszczowe za dni bez deszczu. Te dwie miary razem z accuracy dają dobry obraz działania modelu.