prdom3

April 11, 2021

[87]:
```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier, VotingClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, recall_score,  f1_score,␣
↪precision_score, confusion_matrix
```

[88]:
```
df=pd.read_csv("weatherAUS.csv")
```

[89]:
```
df.info() # Rzeczywiście nie ma nulli w bazie danych
```

```
<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
```

0.1 Analiza danych

Zacznijmy od podzielenia ramki danych względem taretu(*RainTomorrow*) i poszukajmy jakichś zależności między nimi.

[90]:
```
df[df.RainTomorrow==1].describe()
```

[90]:

	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine
count	12427.000000	12427.000000	12427.000000	12427.000000	12427.000000
mean	14.520286	22.285129	5.487302	4.599026	4.529597
std	6.475014	6.866919	11.553907	3.155133	3.390385
min	-4.700000	7.000000	0.000000	0.000000	0.000000
25%	9.200000	16.800000	0.000000	2.200000	1.400000
50%	14.000000	21.400000	0.800000	4.000000	4.300000
75%	19.900000	27.800000	5.800000	6.400000	7.200000
max	29.800000	46.800000	206.200000	43.000000	13.900000

	WindGustSpeed	WindSpeed9am	WindSpeed3pm	Humidity9am	Humidity3pm
count	12427.000000	12427.000000	12427.000000	12427.000000	12427.000000
mean	46.727368	16.980204	21.209624	75.314959	66.905931
std	15.453586	9.031452	9.284455	15.746792	18.449353
min	11.000000	2.000000	2.000000	5.000000	1.000000
25%	35.000000	11.000000	15.000000	66.000000	55.000000
50%	44.000000	15.000000	20.000000	77.000000	68.000000
75%	56.000000	22.000000	28.000000	88.000000	81.000000
max	122.000000	65.000000	65.000000	100.000000	100.000000

	Pressure9am	Pressure3pm	Cloud9am	Cloud3pm	Temp9am
count	12427.000000	12427.000000	12427.000000	12427.000000	12427.000000
mean	1013.926909	1011.816834	5.946729	6.261930	17.980309
std	7.127063	7.163316	2.163760	1.840983	6.599997
min	980.500000	977.100000	0.000000	0.000000	-0.100000
25%	1009.300000	1007.000000	5.000000	6.000000	12.600000
50%	1013.800000	1011.600000	7.000000	7.000000	17.300000
75%	1018.600000	1016.500000	7.000000	7.000000	23.200000
max	1039.500000	1036.000000	8.000000	8.000000	36.400000

	Temp3pm	RainToday	RainTomorrow
count	12427.000000	12427.000000	12427.0
mean	20.348869	0.462139	1.0
std	6.712269	0.498585	0.0
min	4.300000	0.000000	1.0
25%	14.900000	0.000000	1.0
50%	19.600000	0.000000	1.0
75%	25.400000	1.000000	1.0
max	46.100000	1.000000	1.0

[91]:
```
df[df.RainTomorrow==0].describe()
```

[91]:

	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine
count	43993.000000	43993.000000	43993.000000	43993.000000	43993.000000
mean	13.166611	24.765538	1.182149	5.758525	8.641254
std	6.368581	6.902305	4.617274	3.796570	3.338099
min	-6.700000	4.100000	0.000000	0.000000	0.000000
25%	8.400000	19.300000	0.000000	3.000000	6.700000
50%	13.000000	24.600000	0.000000	5.200000	9.500000
75%	18.000000	30.100000	0.200000	7.800000	11.000000
max	31.400000	48.100000	182.600000	81.200000	14.500000

	WindGustSpeed	WindSpeed9am	WindSpeed3pm	Humidity9am	Humidity3pm
count	43993.000000	43993.000000	43993.000000	43993.000000	43993.000000
mean	39.224877	15.296343	19.384857	63.207306	44.714023
std	12.174093	8.065273	8.234055	18.363871	17.847462
min	9.000000	2.000000	2.000000	0.000000	0.000000
25%	31.000000	9.000000	13.000000	53.000000	32.000000
50%	37.000000	15.000000	19.000000	64.000000	46.000000
75%	46.000000	20.000000	24.000000	76.000000	58.000000
max	124.000000	67.000000	76.000000	100.000000	100.000000

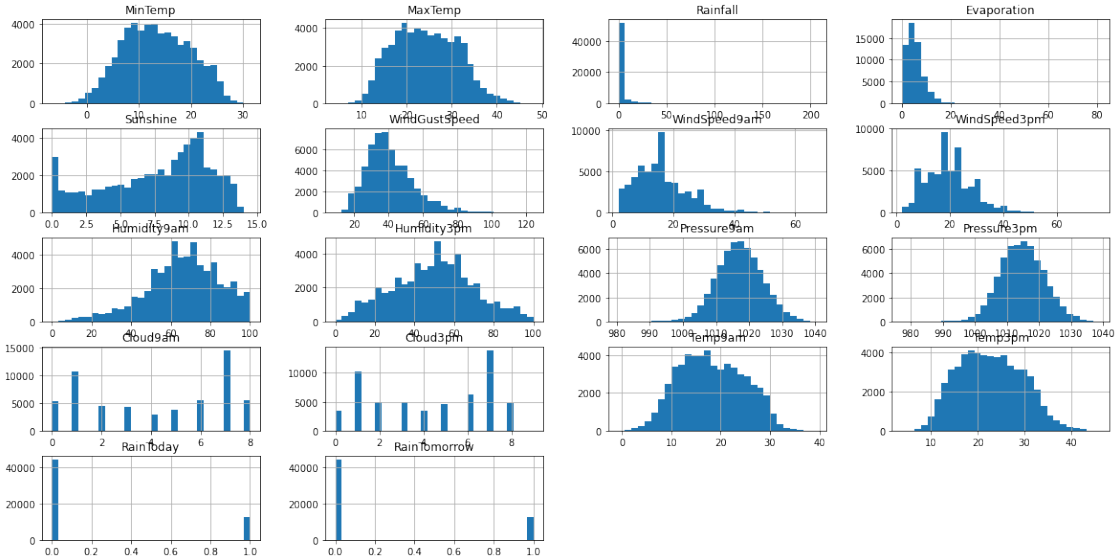
	Pressure9am	Pressure3pm	Cloud9am	Cloud3pm	Temp9am
count	43993.000000	43993.000000	43993.000000	43993.000000	43993.000000
mean	1018.175237	1015.637006	3.760075	3.779806	18.268420
std	6.549944	6.544964	2.767432	2.583256	6.557603
min	982.900000	983.200000	0.000000	0.000000	-0.700000
25%	1013.700000	1011.000000	1.000000	1.000000	13.300000
50%	1018.000000	1015.500000	3.000000	3.000000	17.900000
75%	1022.500000	1020.100000	7.000000	6.000000	23.300000
max	1040.400000	1038.900000	8.000000	9.000000	39.400000

	Temp3pm	RainToday	RainTomorrow
count	43993.000000	43993.000000	43993.0
mean	23.377392	0.152729	0.0
std	6.722703	0.359730	0.0
min	3.700000	0.000000	0.0
25%	18.100000	0.000000	0.0
50%	23.100000	0.000000	0.0
75%	28.500000	0.000000	0.0
max	46.100000	1.000000	0.0

Na pierwszy rzut oka najważniejszymi zmiennymi będą zachmurzenie, wilgotność oraz opady w dniu dzisiejszym, co raczej nikogo nie zdziwi. Co ciekawe ciśnienie spada, a pędkość wiatru rośnie w dni deszczowe.

[92]:
```
df.hist(bins=30,figsize=(20,10))
```

[92]:
```
array([[<AxesSubplot:title={'center':'MinTemp'}>,
        <AxesSubplot:title={'center':'MaxTemp'}>,
        <AxesSubplot:title={'center':'Rainfall'}>,
        <AxesSubplot:title={'center':'Evaporation'}>],
       [<AxesSubplot:title={'center':'Sunshine'}>,
        <AxesSubplot:title={'center':'WindGustSpeed'}>,
        <AxesSubplot:title={'center':'WindSpeed9am'}>,
        <AxesSubplot:title={'center':'WindSpeed3pm'}>],
       [<AxesSubplot:title={'center':'Humidity9am'}>,
        <AxesSubplot:title={'center':'Humidity3pm'}>,
        <AxesSubplot:title={'center':'Pressure9am'}>,
        <AxesSubplot:title={'center':'Pressure3pm'}>],
       [<AxesSubplot:title={'center':'Cloud9am'}>,
        <AxesSubplot:title={'center':'Cloud3pm'}>,
        <AxesSubplot:title={'center':'Temp9am'}>,
        <AxesSubplot:title={'center':'Temp3pm'}>],
       [<AxesSubplot:title={'center':'RainToday'}>,
        <AxesSubplot:title={'center':'RainTomorrow'}>, <AxesSubplot:>,
        <AxesSubplot:>]], dtype=object)
```



1 Modelowanie

[93]:
```
X=df.drop(['RainTomorrow'],axis=1)
y=df['RainTomorrow']
```

[94]:
```
x_train,x_test,y_train,y_test=train_test_split(X,y,test_size=0.
↪3,random_state=152)
```

1.1 Logistic Regression

Zaczniemy od regresji logistycznej z prametrami: - penalty=l2, ponieważ chcemy aby model był zależny od jak najmniejszej liczby parametrów, - dual=False, ponieważ n_samples>n_features, - n_jobs=-1, ponieważ jeśli coś da się zrobić szybciej to nie ma co się oganiczać, - class_weight=balanced, ponieważ dni bez opadów jest sporo więcej, więc chcemy trochę bardziej to zrównoważyć tą dysproporcję.

[95]:
```
model=LogisticRegression(penalty='l2',dual=False,n_jobs=-1,max_iter=1500,random_state=72,class
```

[96]:
```
model.fit(x_train,y_train)
```

[96]:
```
LogisticRegression(class_weight='balanced', max_iter=1500, n_jobs=-1,
                   random_state=72)
```

[97]:
```
y_pred=model.predict(x_test)
```

[98]:
```
print(f"accurancy: {accuracy_score(y_test,y_pred)}")
print(f"recall score: {recall_score(y_test,y_pred)}")
print(f"f1 score: {f1_score(y_test,y_pred)}")
print(f"precision score: {precision_score(y_test,y_pred)}")
confusion_matrix(y_test,y_pred)
```

```
accurancy: 0.7983575564220725
recall score: 0.7923526287838556
f1 score: 0.636179511779128
precision score: 0.531433659839715
```

[98]:
```
array([[10529,  2631],
       [  782,  2984]], dtype=int64)
```

2 RandomForestClassifier

Następnie sprawdzimy Las losowy z prametrami: - n_estimators=100, - min_samples_leaf=5.

[99]:
```
model3=RandomForestClassifier( n_estimators=100, random_state=10,␣
↪min_samples_leaf=5)
```

[100]:
```
model3.fit(x_train,y_train)
y_pred3=model3.predict(x_test)
```

[101]:
```
print(f"accurancy: {accuracy_score(y_test,y_pred3)}")
print(f"recall score: {recall_score(y_test,y_pred3)}")
print(f"f1 score: {f1_score(y_test,y_pred3)}")
print(f"precision score: {precision_score(y_test,y_pred3)}")
confusion_matrix(y_test,y_pred3)
```

```
accurancy: 0.8609831029185868
recall score: 0.5326606479022836
f1 score: 0.6303220738413198
precision score: 0.7718353212774144
```

[101]:
```
array([[12567,   593],
       [ 1760,  2006]], dtype=int64)
```

3 VotingClassifier

Na koniec sprawdzimy VotingClassifier z prametrami: - estimators=[('LR',model),('RFC',model3)], ponieważ chcemy sprawdzić, czy Random Forest połączony z Regresją liniową dadzą wspólnie lepszy efekt, - voting='soft', - n_jobs=-1, ponieważ jeśli coś da się zrobić szybciej to nie ma co się oganiczać.

[115]:
```
model2=VotingClassifier( estimators=[('LR',model),('RFC',model3)],␣
↪voting='soft',n_jobs=-1)
```

[116]:
```
model2.fit(x_train,y_train)
y_pred2=model2.predict(x_test)
```

[117]:
```
print(f"accurancy: {accuracy_score(y_test,y_pred2)}")
print(f"recall score: {recall_score(y_test,y_pred2)}")
print(f"f1 score: {f1_score(y_test,y_pred2)}")
print(f"precision score: {precision_score(y_test,y_pred2)}")
confusion_matrix(y_test,y_pred2)
```

```
accurancy: 0.8460947654496042
recall score: 0.6853425385023898
f1 score: 0.6646066692416635
precision score: 0.6450887278180455
```

[117]:
```
array([[11740,  1420],
       [ 1185,  2581]], dtype=int64)
```

Wybranie najlepszego modelu jest dość subiektywne ze względu na subiektywność, która metryka jest najlepsza dla danego zadania. Jeśli patrzymy na f1 to wygrywa VotingClassifier, najlepsze accurancy osiąga Random Forest, w recall zwycięża regresja logistyczna, a w precision score Random Forest.

[ ]: