NaiveBayes

May 26, 2021

1 Naive Bayes Klassifikator

Als einführendes Beispiel wollen wir mit Hilfe des Naive-Bayes Klassifikators Obsorten, Äpfel und Birnen, andhand des Gewichts und Zuckergehalts klassifizieren.

Wir laden die Daten in ein Dataframe: Zuckergehalt und Gewicht von Äpfeln und Birnen:

[15]:
```
import pandas as pd

url="https://raw.githubusercontent.com/troescherw/datasets/master/obst.csv"
df = pd.read_csv(url, delimiter=";")
df
```

[15]:

	Zuckergehalt	Gewicht	Obstsorte
0	12.0	112	Apfel
1	10.0	100	Apfel
2	9.0	120	Apfel
3	12.0	119	Apfel
4	11.0	115	Apfel
5	13.0	113	Apfel
6	12.0	114	Apfel
7	15.0	150	Birne
8	16.0	149	Birne
9	14.0	147	Birne
10	13.6	151	Birne
11	15.0	150	Birne
12	14.7	149	Birne
13	13.0	140	Birne

Das Feature *Obstsorte* ist nominal skaliert. Deshalb ersetzen wir die String-Wert durch Dummys (One-Hot-Encoding):

0 steht nun für **Apfel**, **1** für **Birne**. Beim **Naive Bayes Klassifikator** ist es übrigens nicht zwingend erforderlich (wie bei OLS), eine Kategorie zu entfernen, hab´s hier aber dennoch mal durchgeführt. Wir können nun unser Modell erstellen.

[16]:
```
from sklearn.naive_bayes import GaussianNB

X = df[["Zuckergehalt", "Gewicht"]]
```

```
y = df.Obstsorte
model = GaussianNB().fit(X,y)
```

Mit Hilfe des Modells können wir nun zwei “unbekannte” Obststücke klassifiziert werden. Haben wir ein Stück Obst, das z.B. ein Zuckergehalt von 52,5g und ein Gewicht von 125g verfügt, fragen wir das Modell, ob es sich um einen Apfel oder eine Birne handelt:

[18]:
```
unbekanntesObst = pd.DataFrame({"Zuckergehalt": [11.5], "Gewicht":[110]})
model.predict(unbekanntesObst)
```

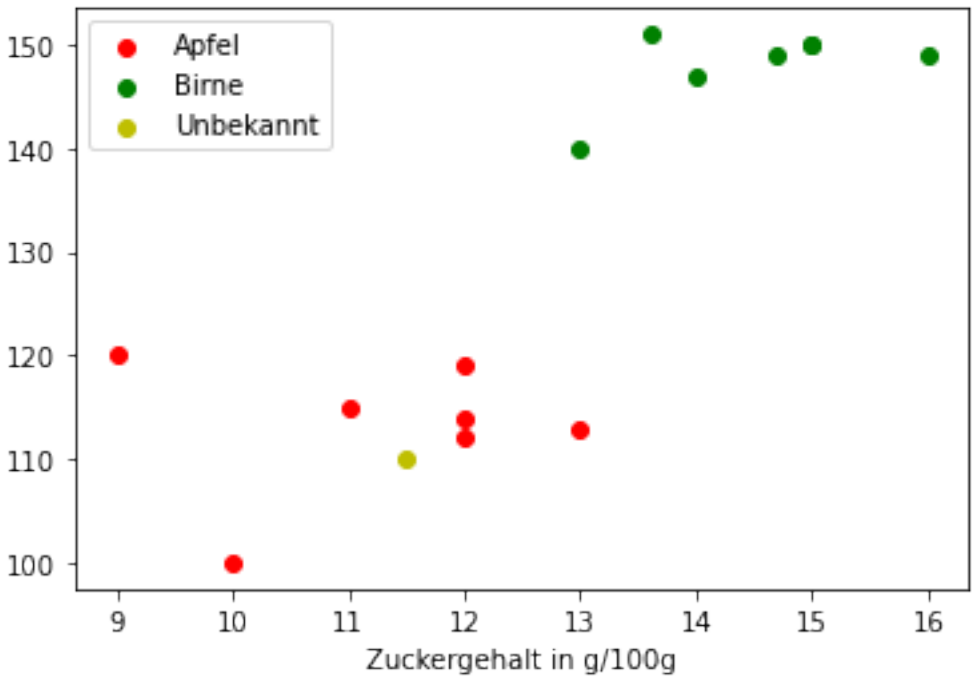
[18]:
```
array(['Apfel'], dtype='<U5')
```

Das Modell gibt eine **0** zurück, damit handelt es sich um einen Apfel. Wir erstellen ein Scatterplot und fügen dort auch das unbekannte Stück Obst ein:

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

plt.scatter(df[df.Obstsorte=="Apfel"].Zuckergehalt, df[df.Obstsorte=="Apfel"].
 ↪Gewicht, c="r", label="Apfel")
plt.scatter(df[df.Obstsorte=="Birne"].Zuckergehalt, df[df.Obstsorte=="Birne"].
 ↪Gewicht, c="g", label="Birne")
plt.scatter(unbekanntesObst.Zuckergehalt, unbekanntesObst.Gewicht, c="y",␣
 ↪label="Unbekannt")
plt.xlabel("Zuckergehalt in g/100g")
plt.legend()
plt.plot()
```

[19]:
```
[]
```



1.1 Beispiel: Naive-Bayes-Modell für die Vorhersage von Brustkrebs

Wir erstellen anhand des bereits verwendeten Datensatzes *breast_cancer* voraus, ob eine Patientin anhand der vorliegenden Daten an Brustkrebs erkrankt ist. Um die Modellqualität zu prüfen führen wir eine k-Fold-Cross-Validation durch (mit k=10).

Wir laden dazu den Datensatz, der von sklearn stammt, und geben die Beschreibung aus:

[6]:
```
from sklearn.datasets import load_breast_cancer
bc = load_breast_cancer()
print(bc.DESCR)
```

```
.. _breast_cancer_dataset:

Breast cancer wisconsin (diagnostic) dataset
--------------------------------------------

**Data Set Characteristics:**

    :Number of Instances: 569

    :Number of Attributes: 30 numeric, predictive attributes and the class

    :Attribute Information:
        - radius (mean of distances from center to points on the perimeter)
        - texture (standard deviation of gray-scale values)
        - perimeter
        - area
        - smoothness (local variation in radius lengths)
        - compactness (perimeter^2 / area - 1.0)
        - concavity (severity of concave portions of the contour)
        - concave points (number of concave portions of the contour)
        - symmetry
        - fractal dimension ("coastline approximation" - 1)

        The mean, standard error, and "worst" or largest (mean of the three
        worst/largest values) of these features were computed for each image,
        resulting in 30 features.  For instance, field 0 is Mean Radius, field
        10 is Radius SE, field 20 is Worst Radius.

        - class:
                - WDBC-Malignant
                - WDBC-Benign

    :Summary Statistics:
```

	Min	Max
radius (mean):	6.981	28.11
texture (mean):	9.71	39.28
perimeter (mean):	43.79	188.5
area (mean):	143.5	2501.0
smoothness (mean):	0.053	0.163
compactness (mean):	0.019	0.345
concavity (mean):	0.0	0.427
concave points (mean):	0.0	0.201
symmetry (mean):	0.106	0.304
fractal dimension (mean):	0.05	0.097
radius (standard error):	0.112	2.873
texture (standard error):	0.36	4.885
perimeter (standard error):	0.757	21.98
area (standard error):	6.802	542.2
smoothness (standard error):	0.002	0.031
compactness (standard error):	0.002	0.135
concavity (standard error):	0.0	0.396
concave points (standard error):	0.0	0.053
symmetry (standard error):	0.008	0.079
fractal dimension (standard error):	0.001	0.03
radius (worst):	7.93	36.04
texture (worst):	12.02	49.54
perimeter (worst):	50.41	251.2
area (worst):	185.2	4254.0
smoothness (worst):	0.071	0.223
compactness (worst):	0.027	1.058
concavity (worst):	0.0	1.252
concave points (worst):	0.0	0.291
symmetry (worst):	0.156	0.664
fractal dimension (worst):	0.055	0.208

:Missing Attribute Values: None

:Class Distribution: 212 - Malignant, 357 - Benign

:Creator: Dr. William H. Wolberg, W. Nick Street, Olvi L. Mangasarian

:Donor: Nick Street

:Date: November, 1995

This is a copy of UCI ML Breast Cancer Wisconsin (Diagnostic) datasets.
https://goo.gl/U2Uwz2

```
Features are computed from a digitized image of a fine needle
aspirate (FNA) of a breast mass.  They describe
characteristics of the cell nuclei present in the image.

Separating plane described above was obtained using
Multisurface Method-Tree (MSM-T) [K. P. Bennett, "Decision Tree
Construction Via Linear Programming." Proceedings of the 4th
Midwest Artificial Intelligence and Cognitive Science Society,
pp. 97-101, 1992], a classification method which uses linear
programming to construct a decision tree.  Relevant features
were selected using an exhaustive search in the space of 1-4
features and 1-3 separating planes.

The actual linear program used to obtain the separating plane
in the 3-dimensional space is that described in:
[K. P. Bennett and O. L. Mangasarian: "Robust Linear
Programming Discrimination of Two Linearly Inseparable Sets",
Optimization Methods and Software 1, 1992, 23-34].

This database is also available through the UW CS ftp server:

ftp ftp.cs.wisc.edu
cd math-prog/cpo-dataset/machine-learn/WDBC/

.. topic:: References

   - W.N. Street, W.H. Wolberg and O.L. Mangasarian. Nuclear feature extraction
     for breast tumor diagnosis. IS&T/SPIE 1993 International Symposium on
     Electronic Imaging: Science and Technology, volume 1905, pages 861-870,
     San Jose, CA, 1993.
   - O.L. Mangasarian, W.N. Street and W.H. Wolberg. Breast cancer diagnosis and
     prognosis via linear programming. Operations Research, 43(4), pages
570-577,
     July-August 1995.
   - W.H. Wolberg, W.N. Street, and O.L. Mangasarian. Machine learning
techniques
     to diagnose breast cancer from fine-needle aspirates. Cancer Letters 77
(1994)
     163-171.
```

[7]:
```
# Wir laden die Daten in X und y
X = bc.data
y = bc.target
```

Wir splitten in X und y auf.

Wir teilen die Datensätze in 10 Folds auf.

[8]:
```
from sklearn.model_selection import KFold
kf = KFold(n_splits=10, shuffle=True)
```

Nun iterieren wir durch die Folds und trainieren das Modell jeweils mit den Daten in den Folds:

[9]:
```
from sklearn.naive_bayes import GaussianNB
model = GaussianNB()

from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X,y,
                                                    shuffle=True,
                                                    random_state=1,
                                                    test_size=0.3)

scores = [] # Leere Liste für Scores

for index_train, index_test in kf.split(X):
    X_train = X[index_train]
    X_test = X[index_test]
    y_train = y[index_train]
    y_test = y[index_test]
    model.fit(X_train, y_train)
    scores.append(model.score(X_test, y_test))

scores
```

[9]:
```
[0.9298245614035088,
 0.9649122807017544,
 0.9122807017543859,
 0.9473684210526315,
 0.9473684210526315,
 0.9298245614035088,
 0.8947368421052632,
 0.9473684210526315,
 0.9824561403508771,
 0.8928571428571429]
```

[10]:
```
from statistics import mean
print(f"Die mittlere Accuracy betägt {mean(scores)}")
```

```
Die mittlere Accuracy betägt 0.9348997493734336
```

[ ]:
```

```