NaiveBayes

June 5, 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:

[ ]:
```
import pandas as pd
import numpy as np

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

Wir können nun unser Modell erstellen.

[ ]:
```
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:

[ ]:
```
unbekanntesObst = pd.DataFrame({"Zuckergehalt": [11.5, 15.1], "Gewicht":[110,␣
↪135]})

print(unbekanntesObst)
model.predict(unbekanntesObst)
```

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:

[ ]:
```
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()
```

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

Wir erstellen anhand des bereits verwendeten Datensatzes *breast_cancer* eine Prognose, ob eine Patientin anhand der vorliegenden Daten an Brustkrebs erkrankt ist (gutartiges oder bösartiges Melanom). Um die Modellqualität zu prüfen führen wir eine k-Fold-Cross-Validation durch (mit k=10). Das heißt, wir erstellen jeweils Testdatensätze (mit 10% der Daten), um die Modellqalität zu prüfen. Es handelt sich jeweils um unterschiedliche Testdatensätze.

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

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

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

[ ]:
```
from sklearn.naive_bayes import GaussianNB
from sklearn.model_selection import KFold

kf = KFold(n_splits=10, shuffle=True)

model = GaussianNB()

from sklearn.model_selection import train_test_split

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
```

[ ]:
```
print(f"Mittelwert Accuracy:  {np.mean(scores)}")
print(f"Standardabweichung der Accuracy: {np.std(scores)}")
```

1.2 Beispiel: Ziffernerkennung

Im Package *sklearn.datasets* befindet sich ein Datensatz *digits*, der Graustufenbilder von handschriftlich erstellten Ziffern 0..9 enthält. Wir versuchen nun mit Hilfe eines Naiven Bayes Klassifikators diese Graustufenbilder den richtigen Klassen (0..9) zuzuordnen.

Wir laden den Datensatz und geben die Beschreibung aus.

[ ]:
```
from sklearn.datasets import load_digits
digits = load_digits()
print(digits.DESCR)
```

Wir geben exemplarisch die ersten 5 Bitmap-Bilder aus:

[ ]:
```
import matplotlib.pyplot as plt
for i in range(5):
    plt.gray()
    plt.matshow(digits.images[i])
    plt.show()
```

Es werden wie üblich die Train- Testdatasets erstellt und das Modell und eine Prediction erstellt. Wir geben die Accuracy und die Confusion Matrix aus.

[ ]:
```
from sklearn.metrics import plot_confusion_matrix
digits = load_digits()

X = digits.data
y = digits.target

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,␣
 ↪shuffle=True, random_state=42)

model = GaussianNB().fit(X_train, y_train)
pred = model.predict(X_test)

print(model.score(X_test, y_test))
plot_confusion_matrix(model, X_test, y_test)
```