Merkblatt

Bestimtheitsmaß

Das Bestimtheitsmaß gibt die Genauigkeit der Formel bzw. der angewandten Methode an, zum Beispiel einer Linearen Regression. Mit dem Maß können verschiedene Modell und Methoden bzgl. der Genauigkeit Ihrer Vorhersage verglichen werden.

[1]:
```
import numpy as np
import pandas as pd
import seaborn as sns

from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split

df = pd.read_csv("./diamonds.csv")

df.head()
```

[1]:

	carat	cut	color	clarity	depth	table	price	x	y	z
0	0.23	Ideal	E	SI2	61.5	55.0	326	3.95	3.98	2.43
1	0.21	Premium	E	SI1	59.8	61.0	326	3.89	3.84	2.31
2	0.23	Good	E	VS1	56.9	65.0	327	4.05	4.07	2.31
3	0.29	Premium	I	VS2	62.4	58.0	334	4.20	4.23	2.63
4	0.31	Good	J	SI2	63.3	58.0	335	4.34	4.35	2.75

Weil der Preis eines Diamanten nicht allein von seiner Karatzahl abhängt, werden im folgenden Beipiel in `xs` mehrere Features angegeben, um dadurch mit mehr Informationen zum jeweiligen Diamanten eine bessere Vorhersage des erzielbaren Preises zu erreichen. Das zeigt sich dann in einem höheren Wert des Bestimmtheitsmaßes. (Achtung! Es dürfen hierfür nur Features gewählt werden, die Integer-Werte oder Float-Werte enthalten.)

Die Methode `.score(xs, ys)` gibt dann das Bestimmtheitsmaß aus.

[2]:
```
xs = df[["carat", "x", "y", "z", "table", "depth"]]
ys = df["price"]
model = LinearRegression()
model.fit(xs, ys)

print(model.score(xs, ys))
```

```
0.8592186831580987
```

Beim Bestimmtheitsmaß gilt:

- 1: Das Modell ist perfekt, jeder Wert wird perfekt beschrieben

- 0: Das Modell ist so gut wie ein “mittlerer Schätzer”
- Negative Werte: Das Modell ist schlechter als ein “mittlerer Schätzer”

Das Vergleichsmodell (hier “mittlerer Schätzer”) genannt, ist ein Modell welches immer den Durchschnitt schätzt. Hier wäre das also ein Modell, welches immer `np.mean(df["price"])` schätzt - und zwar egal wie schwer oder wie groß der Diamant ist.

Mit diesem Vergleichsmodell wird jetzt also unsere Lineare Regression verglichen. Ist sie besser als das Vergleichsmodell (ist im Normalfall der Fall), kommen für das Bestimmtheitsmaß Werte zwischen 0 und 1 heraus.

Das können wir verwenden, um Modelle zu vergleichen, beispielsweise können wir ein Modell nur mit einer Spalte trainieren:

[3]:
```
xscarat = df[["carat"]]
yscarat = df["price"]
model = LinearRegression()
model.fit(xscarat, yscarat)

print(model.score(xscarat, yscarat))
```

```
0.8493305264354857
```

Wir sehen, dass die zusätzlichen Spalten “x”, “y”, “z”, “table”, und “depth” also nur einen minimalen Unterschied auf das Bestimmtheitsmaß machen. Anders ausgedrückt: Wir können ein fast so gutes Ergebnis auch nur mit einer Spalte aus unseren Daten erzeugen.

Trainings- und Test-Daten

Es steht nur eine Tabelle mit Werte zu Diamanten zur Verfügung. Weitere Tabelle dieser Art liege nicht vor. Um dennoch die Genauigkeit eines Modells prüfen zu können, teilen wir nachfolgend das Dataframe in zwei Teile. W

Mit dem ersten Dataframe (`x_train` und `y_train`) trainieren wir das Model mit der in `train_size` festgelegten, zufälligen Prozentsatz der Diamanten auf, hier 75 %. (`random_state` ist ein beliebiger Wert mit dessen Hilfe sich das Ergebniss reproduzieren lässt). Mit den anderen 25 % Testdaten (`x_test` und `y_test`) überprüfen wir die Genauigkeit der Vorhersagen.

Je weniger Trainingsdaten wir dem Model zur Verfügung stellen, desto ungenauer wird die Vorhersage.

[4]:
```
X = df[["carat", "x", "y", "z", "table", "depth"]]
y = df["price"]

X_train, X_test, y_train, y_test = train_test_split(X,y, train_size = 0.75,␣
 ↪random_state = 2)
```

[5]:
```
model.fit(X_train, y_train)

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

```
0.8588260171430935
```