20220120_Informationsmanagement_FAT1

January 20, 2022

[1]:
```
# Mehrfache Regression
```

[2]:
```
# Die mehrfache Regression ist ist wie die lineare Regression ABER
# mit mehr als einen unabhängigen Wert,
# was bedeutet, dass wir versuchen,
# einen Wert basierend auf ZWEI oder MEHR Variablen vorherzusagen.
```

[4]:
```
import pandas

df = pandas.read_csv('/Users/h4/desktop/cars.csv')
```

[5]:
```
df.head()
```

[5]:

	Car	Model	Volume	Weight	CO2
0	Toyoty	Aygo	1000	790	99
1	Mitsubishi	Space Star	1200	1160	95
2	Skoda	Citigo	1000	929	95
3	Fiat	500	900	865	90
4	Mini	Cooper	1500	1140	105

[6]:
```
# Wir erstellen eine LISTE der unabhängigen Werte und nennen Sie diese Variable
 ↪X.

X = df[['Weight', 'Volume']]

# Wir setzen die abhängigen Werte in eine Variable names y

y = df[['CO2']]
```

[8]:
```
# Nun werden wir einige Methoden aus dem "sklearn-Modul" verwenden,
# also müssen wir dieses Modul importieren.

from sklearn import linear_model
```

[9]:
```
# Aus dem sklearn_Modul verwenden wir die LinearRegression() Methode, um ein
 ↪lineares
# Regressionsobjekt zu erstellen.
```

```
# Dieses Objekt hat eine aufgerufene Methode "fit()", die die unabhängigen und␣
 ↪abhängigen Werte
# als Parameter nimmt und das Regressionsobjekt mit Daten füllt, die die␣
 ↪eziehung beschreiben.

regr = linear_model.LinearRegression()
regr.fit(X,y)
```

[9]:
```
LinearRegression()
```

[10]:
```
# Jetzt haben wir ein Regressionsojekt, das bereit ist, CO"-Werte basierend auf␣
 ↪dem Gewicht und Volumen
# eines Autos vorherzusagen.

# Übung. bitte sagen Sie die CO" Werte von einem Fahrzeug mit 2300 Kg Gewicht␣
 ↪und 1300 cm3 Hubraum vor

predictedCO2 = regr.predict([[2300,1300]])

print(predictedCO2)
```

```
[[107.2087328]]
```

```
/Users/h4/opt/anaconda3/lib/python3.8/site-packages/sklearn/base.py:445:
UserWarning: X does not have valid feature names, but LinearRegression was
fitted with feature names
  warnings.warn(
```

[12]:
```
# Koeffizient

# Der Koeffizient ist ein Faktor, der den Zusammenhang mit einer unbekannten␣
 ↪Groesse beschreibt.

# zB. wenn x eine Variable ist, dann 2x ist x zweimal. x ist die unbekannte␣
 ↪Variable und 2 ist der Koeffizient.

# In diesem Fallkönnen wir nach dem Koeffizientenwert von Gewicht zu CO2 und von
# Volumen zu CO2.

# Die Antwort(en), die wir erhalten, sagen uns, was passieren würde, wenn wir␣
 ↪einen der unabhängigen Werte erhöhen oder verringern.

print(regr.coef_)
```

```
[[0.00755095 0.00780526]]
```

[13]:
```
# Ergebnis erklärt:

# Das Ergebnisarray repreäsentiert die Koeffizientenwerte von Gewicht und␣
 ↪Volumen.

# Diese Werte sagen und, dass bei einer Gewichtszunahme von 1kg,
# die CO2 Emission um 0,00755095 gr zunimmt.
```

[14]:
```
#####
```

[15]:
```
# Skalierungsfunktionen
```

[16]:
```
# Wenn Ihre Daten unterscheidliche Werte und sogar unterschiedliche␣
 ↪Masseinheiten haen,
# kann es schwierig sein, sie zu vergleichen.

# Was sind Koligramm im Vergleich zu Metern?

# Die Antwort auf diese Problem ist die Skalierung.
# Wir können Daten in neue Werte skalieren, die einfacher zu vergleichen sind.
```

[18]:
```
import pandas
from sklearn.preprocessing import StandardScaler

# z = (x - Mittelwert(x))/StdAbw(x)

df2 = pandas.read_csv('/Users/h4/desktop/cars2.csv')

df2.head()
```

[18]:

	Car	Model	Volume	Weight	CO2
0	Toyoty	Aygo	1.0	790	99
1	Mitsubishi	Space Star	1.2	1160	95
2	Skoda	Citigo	1.0	929	95
3	Fiat	500	0.9	865	90
4	Mini	Cooper	1.5	1140	105

[20]:
```
scale = StandardScaler()

X2 = df2[['Weight', 'Volume']]

scaledX2 = scale.fit_transform(X2)

print(scaledX2)
```

```
[[-2.10389253 -1.59336644]
 [-0.55407235 -1.07190106]
```

```
[-1.52166278 -1.59336644]
[-1.78973979 -1.85409913]
[-0.63784641 -0.28970299]
[-1.52166278 -1.59336644]
[-0.76769621 -0.55043568]
[ 0.3046118  -0.28970299]
[-0.7551301  -0.28970299]
[-0.59595938 -0.0289703 ]
[-1.30803892 -1.33263375]
[-1.26615189 -0.81116837]
[-0.7551301  -1.59336644]
[-0.16871166 -0.0289703 ]
[ 0.14125238 -0.0289703 ]
[ 0.15800719 -0.0289703 ]
[ 0.3046118  -0.0289703 ]
[-0.05142797  1.53542584]
[-0.72580918 -0.0289703 ]
[ 0.14962979  1.01396046]
[ 1.2219378  -0.0289703 ]
[ 0.5685001   1.01396046]
[ 0.3046118   1.27469315]
[ 0.51404696 -0.0289703 ]
[ 0.51404696  1.01396046]
[ 0.72348212 -0.28970299]
[ 0.8281997   1.01396046]
[ 1.81254495  1.01396046]
[ 0.96642691 -0.0289703 ]
[ 1.72877089  1.01396046]
[ 1.30990057  1.27469315]
[ 1.90050772  1.01396046]
[-0.23991961 -0.0289703 ]
[ 0.40932938 -0.0289703 ]
[ 0.47215993 -0.0289703 ]
[ 0.4302729   2.31762392]]
```

[ ]: