# Logistische_Regression_KlausurBestehen

April 12, 2021

## 1 Logistische Regression

Beispiel: Besteht ein Student / eine Studentin eine Klausur unter Berücksichtigung der Anzahl Stunden für die Vorbereitung auf die Klausur?

Wir laden zuerst den Datensatz.

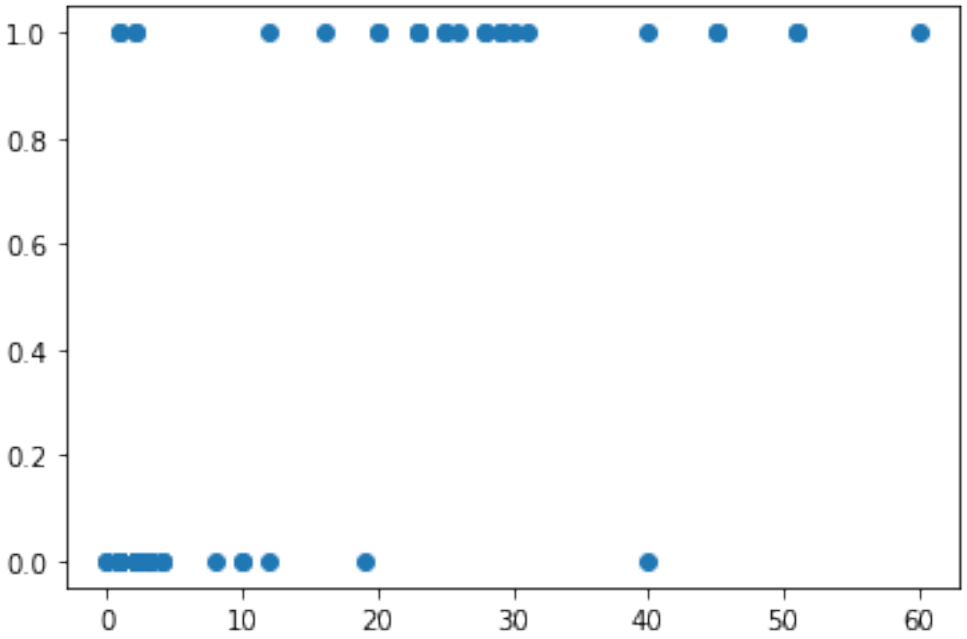
[1]:
```
import pandas as pd
url = "https://raw.githubusercontent.com/troescherw/datasets/master/
 ↪Klausur_Bestanden.csv"
df = pd.read_csv(url, delimiter=";")
df.head()
```

[1]:

| | Stunden | BesuchteVorlesungen | Vorbereitungskurs | KlausurBestanden |
|---|---|---|---|---|
| 0 | 2 | 3 | 0 | 0 |
| 1 | 51 | 12 | 1 | 1 |
| 2 | 19 | 12 | 1 | 0 |
| 3 | 0 | 10 | 0 | 0 |
| 4 | 2 | 1 | 0 | 0 |

Wir erstellen einen Scatterplot.

[2]:
```
%matplotlib inline
import matplotlib.pyplot as plt
plt.scatter(df.Stunden, df.KlausurBestanden)
plt.show()
```




Nun erstellen wir ein Vorhersagemodell mit Hilfe der Logistischen Regression. Wir verwenden wieder das Package *statsmodels.formula.api*. Als *family* geben wir *Binomial* an, da unsere abhängige Variable nur 2 Zustände annehmen kann (0 oder 1). *glm" steht für* Generalized Linear Model*.

[3]:
```
import statsmodels.formula.api as smf
import statsmodels.api as sm

model = smf.glm("KlausurBestanden~Stunden", data=df, family=sm.families.
↪Binomial()).fit()
model.summary()
```

[3]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                 Generalized Linear Model Regression Results
==============================================================================
Dep. Variable:       KlausurBestanden   No. Observations:                   67
Model:                            GLM   Df Residuals:                       65
Model Family:                Binomial   Df Model:                            1
Link Function:                  logit   Scale:                          1.0000
Method:                          IRLS   Log-Likelihood:                -23.982
Date:                Mon, 12 Apr 2021   Deviance:                       47.964
Time:                        10:51:00   Pearson chi2:                     115.
No. Iterations:                     6
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
```

```
------------------------------------------------------------------------------
Intercept     -2.3879      0.531     -4.493      0.000      -3.430      -1.346
Stunden        0.1649      0.037      4.438      0.000       0.092       0.238
==============================================================================
"""
```

Die Koeffizienten lauten:

Für $\beta_0$: -2,39

Für $\beta_1$: 0,165

Damit ergibt sich folgende Formel für die Berechnung von P("Klausur bestanden"):

$P(KlausurBestanden) = \frac{1}{1+e^{2,93-0,165 \cdot Stunden}}$

Wollen wir also zum Beispiel die Wahrscheinlichkeit berechnen, dass eine Studentin eine Klausur besteht, wenn Sie sich 30 Stunden auf die Klausur vorbereitet hat:

$P(KlausurBestanden) = \frac{1}{1+e^{2,39-0,165 \cdot 30}} = 0,93$

Oder wir berechnen diesen Wert mit Hilfe der *pred*-Funktion:

[6]:
```
model.predict({"Stunden":30})
```

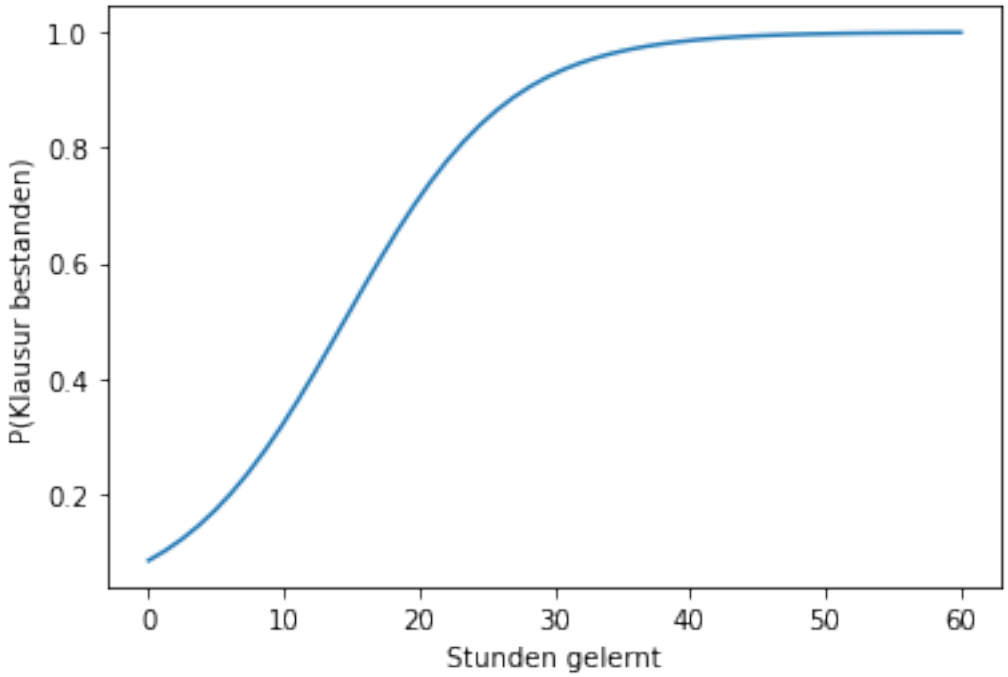
[6]:
```
0    0.928224
dtype: float64
```

Wir plotten noch die "Schwanenhalsfunktion" für die Vorhersagewahrscheinlichkeiten im Bereich der kleinsten und größten Stundenzahl aus dem Datensatz.

[5]:
```
%matplotlib inline
import matplotlib.pyplot as plt
import numpy as np

x = np.linspace(df.Stunden.min(), df.Stunden.max())
y = model.predict(pd.DataFrame({"Stunden":x}))

plt.plot(x,y)
plt.xlabel("Stunden gelernt")
plt.ylabel("P(Klausur bestanden)")
plt.show()
```


1.0
0.8
0.6
0.4
0.2
P(Klausur bestanden)
0
10
20
30
40
50
60
Stunden gelernt