Statistik Rechnerübungen SWB 2 Sommersemester 2022, Aufgabe 2

Nachname: Scholz

Vorname: Noah

Matrikelnummer: 767535

Semester: 2

Email-Adresse: noscit00@hochschule-esslingen.de

Abgabe-Schlusstermin: Donnerstag, 12.05.2022

statistik_labor

May 12, 2022

1 Statistik Labor Aufgabe 2

2 Korrektur:

4.Nachkommastelle des Bestimmtheitsgrades von Polynom 6. richtig gestellt durch ersetzen von 'X' mit 'x_scaled'. Prognose für 2020 ist die selbe geblieben, beim abgleichen mit den Ergebnissen aus 'Teilergebniss_35.txt' stimmen alle gegebenen Nachkommastellen überein. Ein Rundungsfehler konnte nicht gefunden werden. Eigentlich findet auch zu keinem Zeitpunkt, außer beim ausgeben eine Rundung, statt.

Beim absprechen mit Kommilitonen, deren Abgaben schon als Bestanden markiert wurden, wurden auch keine Fehler gefunden.

Mögliche Feedback-Fehlinterpretation: - Fehler liegt tritt nicht in der Berechnung auf sondern bei der Ausgabe (Markdown)

Die Ausgabe, als Markdown wurde folgendermaßen geändert: - Bevölkerungszahl als Ganzzahl statt als Dezimalzahl ausgegeben - Letzte Ziffer auf nächst höhere Ganzzahl kaufmännisch aufgerundet.

Fehler beim berechnen der exponential Funktion gefunden. Werte werden jetzt gefunden.

2.1 Importe und Optionen

[1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import PolynomialFeatures
from sklearn.preprocessing import StandardScaler
from scipy.optimize import curve_fit


pd.set_option('display.max_rows', None)
pd.set_option('display.max_colwidth', 200)
```

2.2 Funktionen

[2]:
```
def readData(file):
    df = pd.read_csv(file, sep="\s+",
                     header=None,
                     names=["jahr","anzahl"],
                     decimal=",",
                     na_values='.',
                     encoding='latin-1')
    return df
```

2.3 Code

2.3.1 Daten einlesen

[3]:
```
data = readData('sr_aufg_2_35.txt')
```

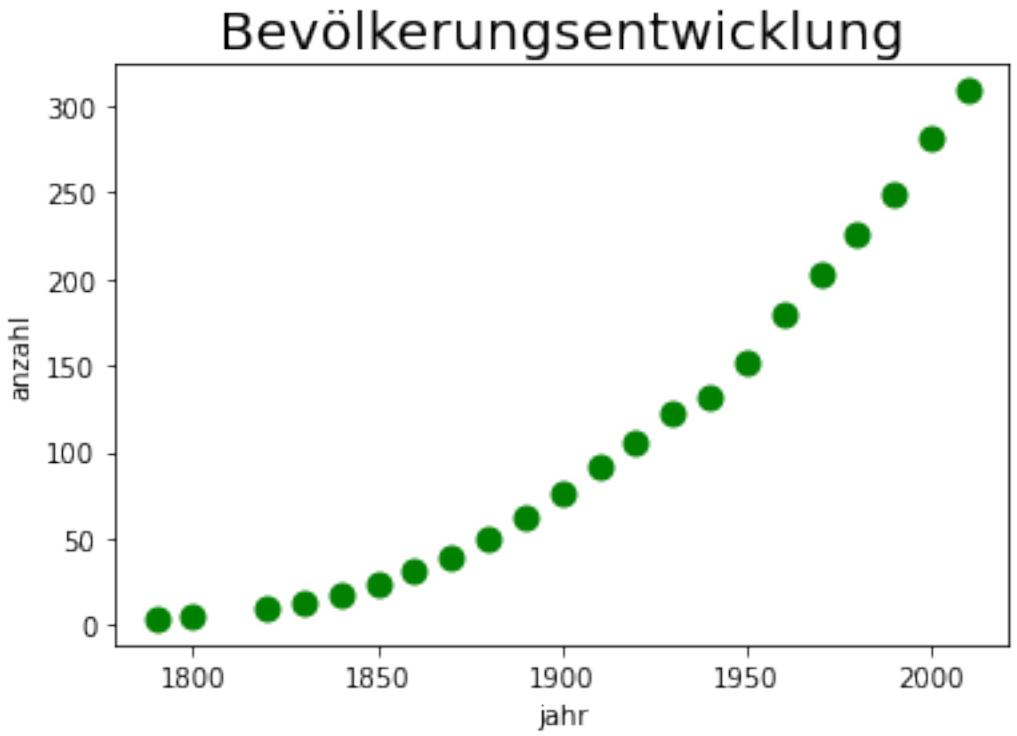
2.3.2 Regressionen durchführen

[4]:
```
X = data['jahr'].to_numpy().reshape(-1, 1)
x_scaled = StandardScaler().fit_transform(X)
linear_regressor = LinearRegression()
linear_regressor.fit(X, data['anzahl'])
r2 = linear_regressor.score(X, data['anzahl'])

# Degree 2
deg_2 = PolynomialFeatures(degree=2, include_bias=False).fit_transform(x_scaled)
poly_regressor_2 = LinearRegression()
poly_regressor_2.fit(deg_2, data['anzahl'])
r2_2 = poly_regressor_2.score(deg_2, data['anzahl'])


# Degree 3
deg_3 = PolynomialFeatures(degree=3, include_bias=False).fit_transform(x_scaled)
poly_regressor_3 = LinearRegression()
poly_regressor_3.fit(deg_3, data['anzahl'])
r2_3 = poly_regressor_3.score(deg_3, data['anzahl'])

# Degree 4
deg_4 = PolynomialFeatures(degree=4, include_bias=False).fit_transform(x_scaled)
poly_regressor_4 = LinearRegression()
poly_regressor_4.fit(deg_4, data['anzahl'])
r2_4 = poly_regressor_4.score(deg_4, data['anzahl'])

# Degree 5
deg_5 = PolynomialFeatures(degree=5, include_bias=False).fit_transform(x_scaled)
poly_regressor_5 = LinearRegression()
```

```
poly_regressor_5.fit(deg_5, data['anzahl'])
r2_5 = poly_regressor_5.score(deg_5, data['anzahl'])

# Degree 6
deg_6 = PolynomialFeatures(degree=6, include_bias=False).fit_transform(x_scaled)
poly_regressor_6 = LinearRegression()
poly_regressor_6.fit(deg_6, data['anzahl'])
r2_6 = poly_regressor_6.score(deg_6, data['anzahl'])

# exponential
def exponential(x, a, b):
    return a * np.exp(b * x)
val = (data['jahr']-np.mean(data['jahr']))/np.std(data['jahr'])
coef_exp, cov_exp = curve_fit(exponential, val, data['anzahl'])
r2_exp = exponential(val, *coef_exp)

res = data['anzahl'] - r2_exp
squared_sum = np.sum(res**2)
squared_sum_total = np.sum((data['anzahl']-np.mean(data['anzahl']))**2)
R_squared = 1 - squared_sum/squared_sum_total

# logarithm
def logarithm(x, a, b):
    return a * np.log(x) - b
val = data['jahr']-np.mean(data['jahr'])/np.std(data['jahr'])
coef_log, cov_log = curve_fit(logarithm, val, data['anzahl'])
r2_log = logarithm(val, *coef_log)

res = data['anzahl'] - r2_log
squared_sum_log = np.sum(res**2)
squared_sum_total_log = np.sum((data['anzahl']-np.mean(data['anzahl']))**2)
R_squared_log = 1 - squared_sum_log/squared_sum_total_log
```

2.3.3 Bevölkerungsentwicklung ausgeben

[5]:
```
# Streudiagramm
scatter = data.plot.scatter(x='jahr', y='anzahl', s=80 , color='green')
scatter=plt.title(fontdict={'fontsize': 20}, label="Bevölkerungsentwicklung")
```



2.3.4 Tabelle erstellen

[6]:
```
# Create Table Strings
linear = {
    0: '$a_1$ = %.2f, $a_0$ = %.2f' % (linear_regressor.coef_, linear_regressor.
 ↪intercept_),
    1: r2
}
poly_2 = {
    0: '$a_2$ = %.2f, $a_1$ = %.2f, $a_0$ = %.2f' % (poly_regressor_2.coef_[1],
                                                     poly_regressor_2.coef_[0],
                                                     poly_regressor_2.

 ↪intercept_),
    1: r2_2
}
poly_3 = {
    0: '$a_3$ = %.2f, $a_2$ = %.2f, $a_1$ = %.2f, $a_0$ = %.2f' %␣
 ↪(poly_regressor_3.coef_[2],
                                                                    ␣

 ↪poly_regressor_3.coef_[1],
                                                                    ␣

 ↪poly_regressor_3.coef_[0],
```

```
                                                                        ␣
 ↪poly_regressor_3.intercept_),
    1: r2_3
}
poly_4 = {
    0: '$a_4$ = %.2f, $a_3$ = %.2f, $a_2$ = %.2f, $a_1$ = %.2f, $a_0$ = %.2f' %␣
 ↪(poly_regressor_4.coef_[3],
                                                                        ␣
 ↪    poly_regressor_4.coef_[2],
                                                                        ␣
 ↪    poly_regressor_4.coef_[1],
                                                                        ␣
 ↪    poly_regressor_4.coef_[0],
                                                                        ␣
 ↪    poly_regressor_4.intercept_),
    1: r2_4
}
poly_5 = {
    0: '$a_5$ = %.2f, $a_4$ = %.2f, $a_3$ = %.2f, $a_2$ = %.2f, $a_1$ = %.2f,␣
 ↪$a_0$ = %.2f'
    % (poly_regressor_5.coef_[4],
         poly_regressor_5.coef_[3],
            poly_regressor_5.coef_[2],
            poly_regressor_5.coef_[1],
            poly_regressor_5.coef_[0],
            poly_regressor_5.intercept_),
    1: r2_5
}
poly_6 = {
    0: '$a_6$ = %.2f, $a_5$ = %.2f, $a_4$ = %.2f, $a_3$ = %.2f, $a_2$ = %.2f,␣
 ↪$a_1$ = %.2f, $a_0$ = %.2f'
    % (poly_regressor_6.coef_[5],
            poly_regressor_6.coef_[4],
            poly_regressor_6.coef_[3],
            poly_regressor_6.coef_[2],
            poly_regressor_6.coef_[1],
            poly_regressor_6.coef_[0],
            poly_regressor_6.intercept_),
    1: r2_6
}
exp = {
    0: '$a$ = %.2f, $b$ = %.2f' % (coef_exp[0], coef_exp[1]),
    1: R_squared
}
ln = {
    0: '$a$ = %.2f, $b$ = %.2f' % (coef_log[0], coef_log[1]),
```

```
    1: R_squared_log
}

dat = [
    linear.values(),
    poly_2.values(),
    poly_3.values(),
    poly_4.values(),
    poly_5.values(),
    poly_6.values(),
    exp.values(),
    ln.values()
]
dataf = pd.DataFrame(dat,
        columns=['Koefizienten', 'Bestimmtheitsmaß'],
        index=['Linear', 'Polynom 2', 'Polynom 3',
               'Polynom 4', 'Polynom 5', 'Polynom 6',
               'Exponentialfunktion', 'Logarithmusfunktion'])
```

2.3.5 Tabelle ausgeben

[7]:
```
print(dataf)
```

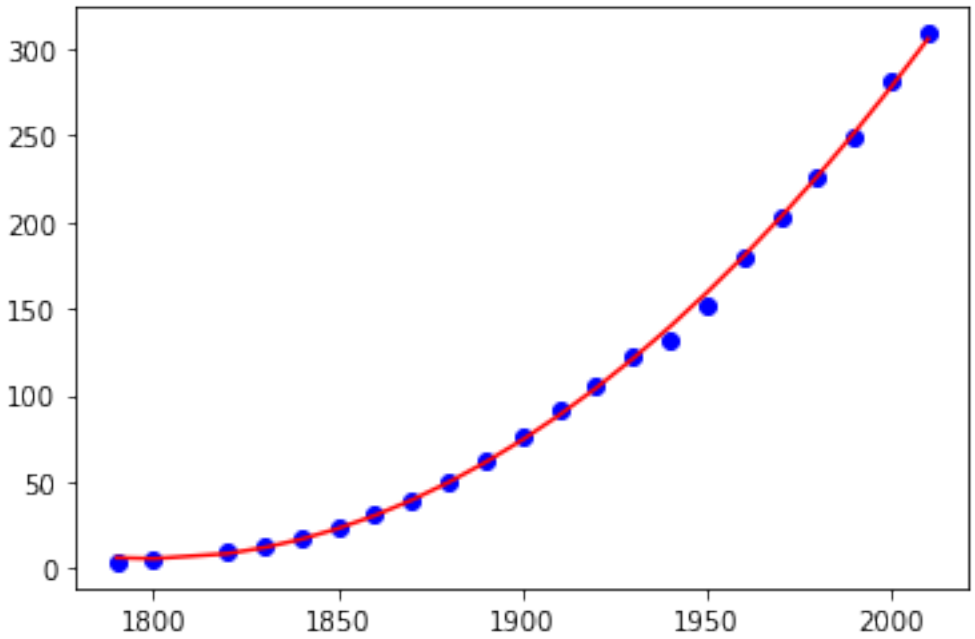
```
                                Koefizienten  \
Linear
$a_1$ = 1.39, $a_0$ = -2530.72
Polynom 2
$a_2$ = 28.48, $a_1$ = 91.93, $a_0$ = 79.96
Polynom 3                                                        $a_3$ = 1.65,
$a_2$ = 28.79, $a_1$ = 88.86, $a_0$ = 79.76
Polynom 4                                          $a_4$ = 1.05, $a_3$ = 1.92,
$a_2$ = 25.99, $a_1$ = 88.54, $a_0$ = 80.62
Polynom 5                             $a_5$ = 0.03, $a_4$ = 1.05, $a_3$ = 1.81,
$a_2$ = 25.97, $a_1$ = 88.62, $a_0$ = 80.63
Polynom 6            $a_6$ = -1.97, $a_5$ = -0.51, $a_4$ = 9.16, $a_3$ = 3.15,
$a_2$ = 17.96, $a_1$ = 87.96, $a_0$ = 81.75
Exponentialfunktion
$a$ = 78.00, $b$ = 0.88
Logarithmusfunktion
$a$ = 2580.48, $b$ = 19337.08

                     Bestimmtheitsmaß
Linear                       0.919293
Polynom 2                    0.999059
Polynom 3                    0.999282
Polynom 4                    0.999348
Polynom 5                    0.999348
```

```
Polynom 6                    0.999467
Exponentialfunktion          0.986139
Logarithmusfunktion          0.910322
```

2.3.6 Regressionskurve zeichnen

Wahl der passenden Kurve Ich habe mich für das Polynom mit dem Grad 2 entschieden, da alle Bestimmtheitsmaße, bis auf die der Logarithmusfunktion und der linearen Kurve, ungefähr gleich sind und das Polynom mit dem Grad zwei die geringste Anzahl an Koeffizienten hat. Außerdem stimmt die Prognose, der Quadratischen funktion, mit den Testergegnissen überein.

[8]:
```
deg_2_pred = poly_regressor_2.predict(deg_2)
plt.scatter(data['jahr'], data['anzahl'], color='blue')
plt.plot(X, deg_2_pred, color='red')
plt.show()
```



2.3.7 Prognose durchführen

[9]:
```
def calculate_2020_poly_2(x, a0, a1, a2):
    return a0 + a1 * x + a2 * x**2


# polynomial 2
poly_2_2020 = calculate_2020_poly_2((2020-np.mean(data['jahr']))/np.
 ↪std(data['jahr']), poly_regressor_2.intercept_, *poly_regressor_2.coef_)
print(poly_2_2020)
```

```
334.85895092496963
```

Die geschätzte Einwohnerzahl im Jahr 2020 beträgt 334.858.951 Menschen.

Ich habe die geschätzten koeffizienten(`a,b`) und die Verschiebung auf der y-Achse(`c`) in ein Polynom 2. Grades

$$y(x) = ax^2 + bx + c$$

Wobei $x :=$ "$Jahr$" und $y(x) :=$ "$Einwohneranzahl$" eingesetzt. Mit $x = 2020$ ist $y(2020)$ die geschätzte Einwohnerzahl im Jahr 2020.