Übung 02

Prognosen und Exponentielles Glätten

Aufgabe 1: Prognose mit Trend und Saison

Luigi's Eisdiele in Duisburg hat saisonal stark schwankende Verkaufszahlen (in hundert Litern Eis). Es gibt vier Quartale pro Jahr. Luigi hat die Verkaufszahlen der letzten **zwei** Jahre gesammelt:

Jahr	Quartal	Verkauf (100L) y_t
1	Q1	8
1	Q2	20
1	Q3	25
1	Q4	10
2	Q1	10
2	Q2	24
2	Q3	30
2	Q4	12

Luigi hat bereits (vereinfachte) **Saisonindizes** für die vier Quartale bestimmt:

- Q1: 0.6
- Q2: 1.4
- Q3: 1.7
- Q4: 0.8

Er geht von einem multiplikativen Saisonmodell aus ($Y = T \cdot S \cdot I$).

Ihre Aufgaben:

1. Bereinigen Sie die Verkaufszahlen, indem Sie jeden Wert durch den entsprechenden Saisonindex teilen. Diese Werte Y_d repräsentieren die Trend-Komponente (plus Rest).
2. Auf die bereinigten Daten Y_d wenden Sie nun eine Prognosemethode für Trenddaten an. Verwenden Sie hierfür die exponentielle Glättung mit Trendkorrektur.
   - Nutzen Sie einen Glättungsfaktor $\alpha = 0.2$.
   - Initialisierungswerte für die bereinigten Daten (Y_d) zum Zeitpunkt $t = 0$:
     - Geschätztes Niveau $\hat{a}_{d,0} = 13$
     - Geschätzter Trend $\hat{b}_{d,0} = 0.5$
   - Berechnen Sie zuerst $y^{(1)}_{d,0}$ und $y^{(2)}_{d,0}$ basierend auf $\hat{a}_{d,0}$, $\hat{b}_{d,0}$ und α.
3. Prognostizieren Sie die bereinigten Werte für die vier Quartale des nächsten Jahres (Jahr 3, Q1 bis Q4). Hier ist t der Index des letzten Beobachtungspunktes der bereinigten Reihe.

4. Saisonalisieren Sie diese Prognosen, indem Sie sie mit den entsprechenden Saisonindizes multiplizieren, um die finalen Verkaufsprognosen zu erhalten.

Lösung (Python-Code):

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

plt.rcdefaults()
plt.style.use('tableau-colorblind10')

data_task4 = {
    'Jahr': [1]*4 + [2]*4,
    'QuartalStr': ['Q1', 'Q2', 'Q3', 'Q4'] * 2, # String für Mapping
    'Verkauf': [8, 20, 25, 10, 10, 24, 30, 12]
}
df_task4 = pd.DataFrame(data_task4)
df_task4['Periode'] = range(1, len(df_task4) + 1)

saison_indizes = {'Q1': 0.6, 'Q2': 1.4, 'Q3': 1.7, 'Q4': 0.8}
df_task4['Saisonindex'] = df_task4['QuartalStr'].map(saison_indizes)

print("Verkaufsdaten Gelato Paradiso:")
print(df_task4)

# 1. Désaisonieren
df_task4['Verkauf_desaisoniert'] = df_task4['Verkauf'] /
df_task4['Saisonindex']
print("\nDésaisonierte Verkaufsdaten (Y_d):")
print(df_task4[['Jahr', 'QuartalStr', 'Verkauf', 'Saisonindex',
'Verkauf_desaisoniert']].round(2))

y_d_actual = df_task4['Verkauf_desaisoniert'].tolist()

# 2. Holt's Methode auf désaisonierte Daten Y_d
alpha_t4 = 0.2
ad0_hat = 13  # Niveau-Schätzung für Y_d bei t=0
bd0_hat = 0.5 # Trend-Schätzung für Y_d bei t=0

# Initialisierung y_d,0^(1), y_d,0^(2)
if alpha_t4 == 0 or alpha_t4 == 1:
    print("Alpha darf nicht 0 oder 1 sein für diese Initialisierung.")
    yd0_1 = np.nan
    yd0_2 = np.nan
else:
    yd0_1 = ad0_hat - bd0_hat * (1 - alpha_t4) / alpha_t4
    yd0_2 = ad0_hat - 2 * bd0_hat * (1 - alpha_t4) / alpha_t4

print(f"\nInitialisierung für désaisonierte Daten Y_d:")
print(f"y_d,0^(1) = {yd0_1:.2f}")
print(f"y_d,0^(2) = {yd0_2:.2f}")
```

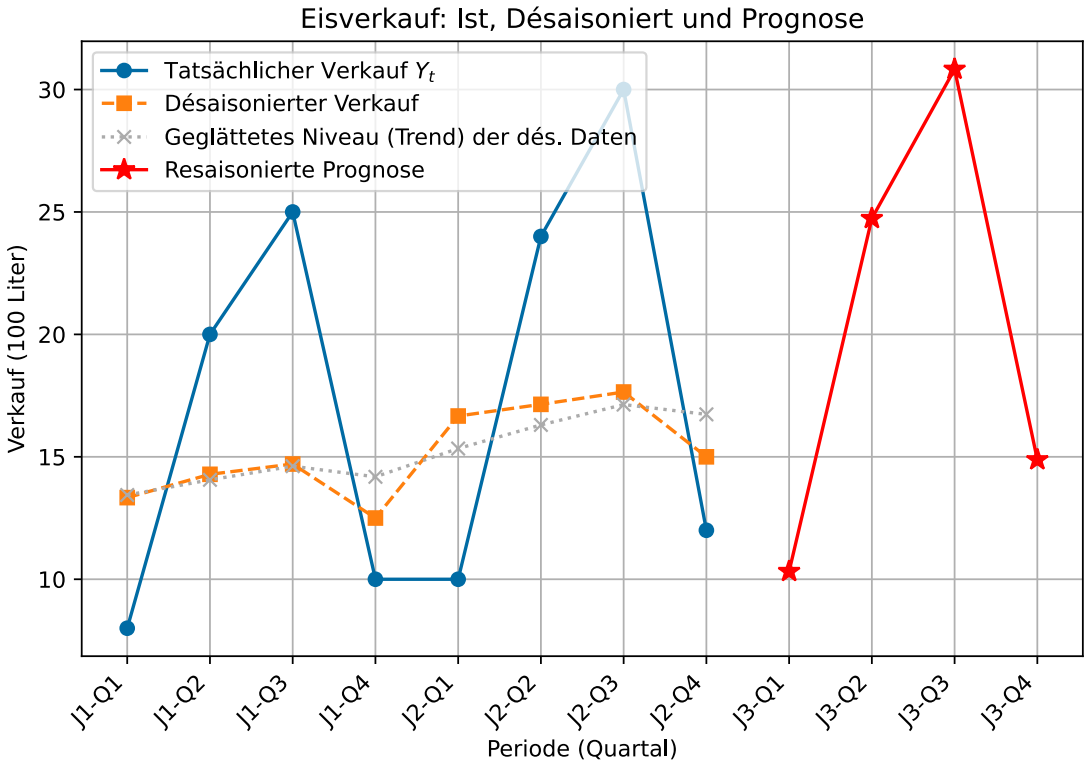
```python
# Arrays für die Speicherung der Werte für Y_d
y1_d_vals = [0.0] * (len(y_d_actual) + 1)
y2_d_vals = [0.0] * (len(y_d_actual) + 1)
a_d_hat_vals = [0.0] * (len(y_d_actual) + 1)
b_d_hat_vals = [0.0] * (len(y_d_actual) + 1)

y1_d_vals[0] = yd0_1
y2_d_vals[0] = yd0_2

results_t4_desais = []
print("\nHolt's Methode auf désaisonierten Daten Y_d (t=1 bis 8):")

for t_idx in range(1, len(y_d_actual) + 1): # t_idx von 1 bis 8
    y_dt = y_d_actual[t_idx-1]

    y1_d_vals[t_idx] = alpha_t4 * y_dt + (1 - alpha_t4) *
y1_d_vals[t_idx-1]
    y2_d_vals[t_idx] = alpha_t4 * y1_d_vals[t_idx] + (1 - alpha_t4) *
y2_d_vals[t_idx-1]

    a_d_hat_vals[t_idx] = 2 * y1_d_vals[t_idx] - y2_d_vals[t_idx]
    if alpha_t4 == 1:
        b_d_hat_vals[t_idx] = float('inf') if (y1_d_vals[t_idx] -
y2_d_vals[t_idx]) != 0 else 0
    else:
        b_d_hat_vals[t_idx] = (alpha_t4 / (1 - alpha_t4)) *
(y1_d_vals[t_idx] - y2_d_vals[t_idx])

    results_t4_desais.append({
        't': t_idx, 'y_d_t': y_dt,
        'y1_d_t': y1_d_vals[t_idx], 'y2_d_t': y2_d_vals[t_idx],
        'a_d_hat_t': a_d_hat_vals[t_idx], 'b_d_hat_t': b_d_hat_vals[t_idx]
    })

df_results_t4_desais = pd.DataFrame(results_t4_desais)
# Add calculated a_d_hat_t and b_d_hat_t to the original df_task4 for
easier plotting if needed
# Merge based on 't' (Periode)
df_results_t4_desais.rename(columns={'t': 'Periode'}, inplace=True)
df_task4 = pd.merge(df_task4, df_results_t4_desais[['Periode', 'a_d_hat_t',
'b_d_hat_t']], on='Periode', how='left')


print(df_results_t4_desais.round(2))

# 3. Prognose der désaisonierten Werte für Jahr 3 (Periode 9, 10, 11, 12)
# Basierend auf a_d_hat_8 und b_d_hat_8
a_d_hat_final = a_d_hat_vals[len(y_d_actual)] # a_d_hat_8
b_d_hat_final = b_d_hat_vals[len(y_d_actual)] # b_d_hat_8

prognosen_desaisoniert_j3 = []
quartale_j3_str = ['Q1', 'Q2', 'Q3', 'Q4']
```

```
print(f"\nLetzte Parameter für Y_d (t=8):")
print(f"a_d_hat_8 = {a_d_hat_final:.2f}")
print(f"b_d_hat_8 = {b_d_hat_final:.2f}")

print("\nPrognostizierte désaisonierte Werte Y_d für Jahr 3:")
for i in range(1, 5): # i = 1, 2, 3, 4 (für Periode 9, 10, 11, 12)
    # p_d,8+i = a_d_hat_8 + b_d_hat_8 * i
    p_di = a_d_hat_final + b_d_hat_final * i
    prognosen_desaisoniert_j3.append({
        'Periode': len(y_d_actual) + i,
        'QuartalStr': quartale_j3_str[i-1],
        'Prognose_Desaisoniert': p_di
    })
    print(f"Jahr 3, {quartale_j3_str[i-1]} (Periode {len(y_d_actual) + i}):
{p_di:.2f}")

df_prognosen_desaisoniert_j3 = pd.DataFrame(prognosen_desaisoniert_j3)

# 4. Resaisonieren
df_prognosen_desaisoniert_j3['Saisonindex'] =
df_prognosen_desaisoniert_j3['QuartalStr'].map(saison_indizes)
df_prognosen_desaisoniert_j3['Prognose_Final'] =
df_prognosen_desaisoniert_j3['Prognose_Desaisoniert'] *
df_prognosen_desaisoniert_j3['Saisonindex']

print("\nFinale Verkaufsprognosen für Jahr 3 (resaisoniert):")
print(df_prognosen_desaisoniert_j3[['Periode', 'QuartalStr',
'Prognose_Desaisoniert', 'Saisonindex', 'Prognose_Final']].round(2))

# Plot für Aufgabe 4
plt.figure(figsize=(7, 5))

# Tatsächliche Verkäufe
plt.plot(df_task4['Periode'], df_task4['Verkauf'], marker='o',
linestyle='-', label='Tatsächlicher Verkauf $Y_t$')

# Désaisonierte Verkäufe
plt.plot(df_task4['Periode'], df_task4['Verkauf_desaisoniert'], marker='s',
linestyle='--', label='Désaisonierter Verkauf')

# Trend der désaisonierten Daten (a_d_hat_t)
plt.plot(df_task4['Periode'], df_task4['a_d_hat_t'], marker='x',
linestyle=':', label='Geglättetes Niveau (Trend) der dés. Daten')

# Resaisonierte Prognosen für Jahr 3
plt.plot(df_prognosen_desaisoniert_j3['Periode'],
df_prognosen_desaisoniert_j3['Prognose_Final'], marker='*', linestyle='-',
color='red', markersize=10, label='Resaisonierte Prognose')

plt.title(f'Eisverkauf: Ist, Désaisoniert und Prognose')
plt.xlabel('Periode (Quartal)')
plt.ylabel('Verkauf (100 Liter)')
```

```python
# X-Ticks für bessere Lesbarkeit
num_total_periods = len(df_task4) + len(df_prognosen_desaisoniert_j3)
tick_positions = list(range(1, num_total_periods + 1))
tick_labels = [f"J{df_task4.loc[p-1, 'Jahr']}-Q{df_task4.loc[p-1,
'QuartalStr'][-1]}" if p <= len(df_task4) else f"J3-
Q{df_prognosen_desaisoniert_j3.loc[p-len(df_task4)-1, 'QuartalStr'][-1]}"
for p in tick_positions]

plt.xticks(tick_positions, tick_labels, rotation=45, ha="right")
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()
```

```
Verkaufsdaten Gelato Paradiso:
   Jahr QuartalStr  Verkauf  Periode  Saisonindex
0     1         Q1        8        1          0.6
1     1         Q2       20        2          1.4
2     1         Q3       25        3          1.7
3     1         Q4       10        4          0.8
4     2         Q1       10        5          0.6
5     2         Q2       24        6          1.4
6     2         Q3       30        7          1.7
7     2         Q4       12        8          0.8

Désaisonierte Verkaufsdaten (Y_d):
   Jahr QuartalStr  Verkauf  Saisonindex  Verkauf_desaisoniert
0     1         Q1        8          0.6                 13.33
1     1         Q2       20          1.4                 14.29
2     1         Q3       25          1.7                 14.71
3     1         Q4       10          0.8                 12.50
4     2         Q1       10          0.6                 16.67
5     2         Q2       24          1.4                 17.14
6     2         Q3       30          1.7                 17.65
7     2         Q4       12          0.8                 15.00

Initialisierung für désaisonierte Daten Y_d:
y_d,0^(1) = 11.00
y_d,0^(2) = 9.00

Holt's Methode auf désaisonierten Daten Y_d (t=1 bis 8):
   Periode  y_d_t  y1_d_t  y2_d_t  a_d_hat_t  b_d_hat_t
0        1  13.33   11.47    9.49      13.44       0.49
1        2  14.29   12.03   10.00      14.06       0.51
2        3  14.71   12.57   10.51      14.62       0.51
3        4  12.50   12.55   10.92      14.18       0.41
4        5  16.67   13.38   11.41      15.34       0.49
5        6  17.14   14.13   11.96      16.30       0.54
6        7  17.65   14.83   12.53      17.13       0.58
7        8  15.00   14.87   13.00      16.73       0.47
```

```
Letzte Parameter für Y_d (t=8):
a_d_hat_8 = 16.73
b_d_hat_8 = 0.47

Prognostizierte désaisonierte Werte Y_d für Jahr 3:
Jahr 3, Q1 (Periode 9): 17.20
Jahr 3, Q2 (Periode 10): 17.67
Jahr 3, Q3 (Periode 11): 18.14
Jahr 3, Q4 (Periode 12): 18.60

Finale Verkaufsprognosen für Jahr 3 (resaisoniert):
   Periode QuartalStr  Prognose_Desaisoniert  Saisonindex  Prognose_Final
0        9         Q1                  17.20          0.6           10.32
1       10         Q2                  17.67          1.4           24.74
2       11         Q3                  18.14          1.7           30.83
3       12         Q4                  18.60          0.8           14.88
```



Aufgabe 2: Saisonindizes selbst berechnen

Das Unternehmen Frosty, das auch die Eisdiele von Luigi beliefert, hat sich auf die Herstellung und den Verkauf von handgemachtem Eis spezialisiert. Die Geschäftsführerin hat festgestellt, dass die Verkaufszahlen (in tausend Euro) stark von der Jahreszeit abhängen. Um die Produktionsmengen besser planen und Marketingkampagnen gezielter ausrichten zu können, möchte sie die saisonalen Schwankungen genauer verstehen. Sie hat die Verkaufsdaten der letzten **drei** Jahre gesammelt:

Jahr	Quartal	Verkauf (Tsd. €) y_t
1	Q1	150
1	Q2	250
1	Q3	350
1	Q4	180
2	Q1	170
2	Q2	280
2	Q3	390
2	Q4	210
3	Q1	190
3	Q2	310
3	Q3	430
3	Q4	240

Die Geschäftsführerin geht von einem multiplikativen Saisonmodell aus ($Y = T \cdot C \cdot S \cdot I$) und möchte die Saisonindizes mit der "Ratio to Moving Average"-Methode bestimmen. Da es vier Quartale pro Jahr gibt, wird ein gleitender Durchschnitt der Ordnung 4 verwendet.

Ihre Aufgaben:

1. Berechnen Sie den zentrierten gleitenden Durchschnitt (ZGD) der Ordnung 4 für die Verkaufsdaten.
2. Bestimmen Sie die Roh-Saisonfaktoren (si_{tm}), indem Sie die tatsächlichen Verkaufszahlen y_{tm} durch die entsprechenden ZGD-Werte teilen.
3. Berechnen Sie die durchschnittlichen Saisonfaktoren (s_m) für jedes Quartal, indem Sie die Roh-Saisonfaktoren für das jeweilige Quartal über die Jahre mitteln.
4. Normieren Sie die durchschnittlichen Saisonfaktoren, sodass ihre Summe der Anzahl der Saisons (hier 4) entspricht. Diese normierten Werte sind die finalen Saisonindizes ($\hat{s}_m$).

Lösung (Python-Code):

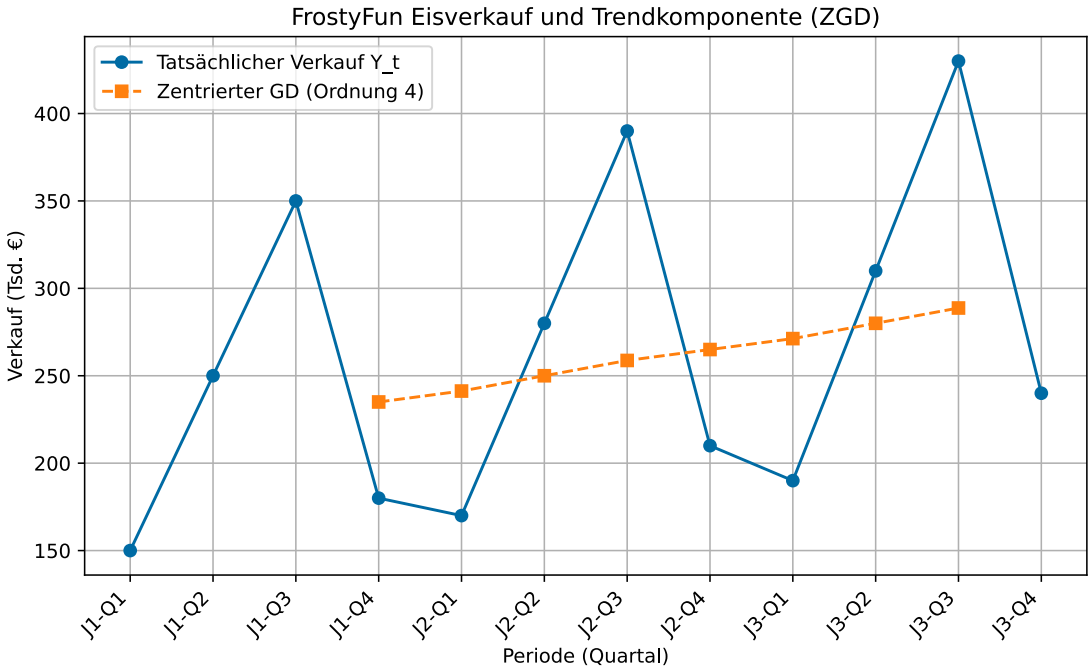
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

plt.rcdefaults()
plt.style.use('tableau-colorblind10')

# Daten für Aufgabe 3
data_task3 = {
    'Jahr': [1]*4 + [2]*4 + [3]*4,
    'QuartalStr': ['Q1', 'Q2', 'Q3', 'Q4'] * 3,
    'Verkauf': [150, 250, 350, 180, 170, 280, 390, 210, 190, 310, 430, 240]
}
df_task3 = pd.DataFrame(data_task3)
```

```python
df_task3['Periode'] = range(1, len(df_task3) + 1)

print("Verkaufsdaten FrostyFun Eiscreme:")
print(df_task3)

# 1. Zentrierter Gleitender Durchschnitt (ZGD) der Ordnung 4
# Einfacher GD der Ordnung 4
df_task3['GD4'] = df_task3['Verkauf'].rolling(window=4).mean()
# Zentrierter GD (Durchschnitt von 2 aufeinanderfolgenden GD4 Werten)
df_task3['ZGD4'] = df_task3['GD4'].rolling(window=2).mean().shift(-1) #
shift(-1) zur korrekten Zuordnung

print("\nVerkaufsdaten mit GD4 und ZGD4:")
print(df_task3[['Periode', 'Verkauf', 'GD4', 'ZGD4']].round(2))

# 2. Roh-Saisonfaktoren (si_tm = y_tm / ZGD_tm)
df_task3['Roh_SF'] = df_task3['Verkauf'] / df_task3['ZGD4']

print("\nVerkaufsdaten mit Roh-Saisonfaktoren:")
# Anzeigen der relevanten Spalten, Zeilen mit NaN in Roh_SF werden erstmal
ignoriert für die Ausgabe
print(df_task3[['Periode', 'QuartalStr', 'Verkauf', 'ZGD4',
'Roh_SF']].dropna().round(3))

# 3. Durchschnittliche Saisonfaktoren (s_m)
# Zuerst Quartalsnummer extrahieren (1, 2, 3, 4)
df_task3['QuartalNum'] = df_task3['QuartalStr'].str[1].astype(int)
durchschnittliche_sf = df_task3.groupby('QuartalNum')
['Roh_SF'].mean().reset_index()
durchschnittliche_sf.rename(columns={'Roh_SF': 's_m_durchschnitt'},
inplace=True)

print("\nDurchschnittliche Saisonfaktoren (s_m):")
print(durchschnittliche_sf.round(3))

# 4. Normierte Saisonindizes (s_dach_m)
summe_s_m = durchschnittliche_sf['s_m_durchschnitt'].sum()
anzahl_saisons = 4
normierungsfaktor = anzahl_saisons / summe_s_m
durchschnittliche_sf['Saisonindex_Normiert'] =
durchschnittliche_sf['s_m_durchschnitt'] * normierungsfaktor

print(f"\nSumme der durchschnittlichen SF: {summe_s_m:.3f}")
print(f"Normierungsfaktor: {normierungsfaktor:.3f}")

print("\nFinale normierte Saisonindizes:")
print(durchschnittliche_sf[['QuartalNum', 's_m_durchschnitt',
'Saisonindex_Normiert']].round(3))
print(f"Summe der normierten Saisonindizes:
{durchschnittliche_sf['Saisonindex_Normiert'].sum():.2f}")
```

```python
# Plot für Aufgabe 3 (Optional, aber hilfreich zur Visualisierung)
plt.figure(figsize=(8, 5))
plt.plot(df_task3['Periode'], df_task3['Verkauf'], marker='o',
linestyle='-', label='Tatsächlicher Verkauf Y_t')
plt.plot(df_task3['Periode'], df_task3['ZGD4'], marker='s', linestyle='--',
label='Zentrierter GD (Ordnung 4)')

plt.title('FrostyFun Eisverkauf und Trendkomponente (ZGD)')
plt.xlabel('Periode (Quartal)')
plt.ylabel('Verkauf (Tsd. €)')

tick_positions = df_task3['Periode'].tolist()
tick_labels = [f"J{df_task3.loc[p-1, 'Jahr']}-Q{df_task3.loc[p-1,
'QuartalStr'][-1]}" for p in tick_positions]
plt.xticks(tick_positions, tick_labels, rotation=45, ha="right")

plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()

# Tabelle der Saisonindizes für die Präsentation
saisonindizes_final =
pd.Series(durchschnittliche_sf['Saisonindex_Normiert'].values, index=['Q1',
'Q2', 'Q3', 'Q4'])
print("\nÜbersicht der finalen Saisonindizes:")
print(saisonindizes_final.round(3))
```

```
Verkaufsdaten FrostyFun Eiscreme:
    Jahr QuartalStr  Verkauf  Periode
0      1         Q1      150        1
1      1         Q2      250        2
2      1         Q3      350        3
3      1         Q4      180        4
4      2         Q1      170        5
5      2         Q2      280        6
6      2         Q3      390        7
7      2         Q4      210        8
8      3         Q1      190        9
9      3         Q2      310       10
10     3         Q3      430       11
11     3         Q4      240       12

Verkaufsdaten mit GD4 und ZGD4:
    Periode  Verkauf    GD4    ZGD4
0         1      150    NaN     NaN
1         2      250    NaN     NaN
2         3      350    NaN     NaN
3         4      180  232.5  235.00
4         5      170  237.5  241.25
5         6      280  245.0  250.00
```

```
6          7      390  255.0  258.75
7          8      210  262.5  265.00
8          9      190  267.5  271.25
9         10      310  275.0  280.00
10        11      430  285.0  288.75
11        12      240  292.5     NaN

Verkaufsdaten mit Roh-Saisonfaktoren:
    Periode QuartalStr  Verkauf    ZGD4  Roh_SF
3          4         Q4      180  235.00   0.766
4          5         Q1      170  241.25   0.705
5          6         Q2      280  250.00   1.120
6          7         Q3      390  258.75   1.507
7          8         Q4      210  265.00   0.792
8          9         Q1      190  271.25   0.700
9         10         Q2      310  280.00   1.107
10        11         Q3      430  288.75   1.489

Durchschnittliche Saisonfaktoren (s_m):
   QuartalNum  s_m_durchschnitt
0           1             0.703
1           2             1.114
2           3             1.498
3           4             0.779

Summe der durchschnittlichen SF: 4.094
Normierungsfaktor: 0.977

Finale normierte Saisonindizes:
   QuartalNum  s_m_durchschnitt  Saisonindex_Normiert
0           1             0.703                 0.687
1           2             1.114                 1.088
2           3             1.498                 1.464
3           4             0.779                 0.761
Summe der normierten Saisonindizes: 4.00
```




```
Übersicht der finalen Saisonindizes:
Q1    0.687
Q2    1.088
Q3    1.464
Q4    0.761
dtype: float64
```

Aufgabe 3: Holt's Methode für Trenddaten

Das kleine Startup "Deep Learning" im Herzen von Duisburg hat kürzlich ein neues KI-Tool für die Analyse von Textdaten auf den Markt gebracht. Die Kunden sind begeistert, und die Verkaufszahlen steigen stetig. Die Firma möchte nun die zukünftige Nachfrage besser planen können, um genügend Mitarbeiter zur Betreuung der Kunden zu haben und gleichzeitig Überstunden zu vermeiden. Die Firma hat die Verkaufszahlen der letzten 8 Monate sorgfältig dokumentiert:

Monat (t)	Verkäufe (y_t)
1	50
2	52
3	58
4	69
5	70
6	72
7	77

Monat (t)	Verkäufe (y_t)
8	83

Deep Learning hat sich entschieden, das **Verfahren von Holt** zu verwenden, um eine Prognose zu erstellen. Dieses Verfahren berücksichtigt sowohl das aktuelle Niveau der Nachfrage als auch den Trend.

Ihre Aufgaben:

1. Verwenden Sie das Verfahren von Holt, um die geglätteten Werte für das Niveau ($\hat{a}_t$) und den Trend ($\hat{b}_t$) für die Monate $t = 1$ bis $t = 8$ zu berechnen.
   - Nutzen Sie die folgenden Glättungsfaktoren:
     - $\alpha = 0.3$ (für das Niveau)
     - $\beta = 0.2$ (für den Trend)
   - Die Initialisierungswerte zum Zeitpunkt $t = 0$ sind:
     - Geschätztes Niveau $\hat{a}_0 = 48$ Verkäufe
     - Geschätzter Trend $\hat{b}_0 = 4$ Verkäufe pro Monat
2. Erstellen Sie eine Prognose für die Verkaufszahlen der nächsten **zwei** Monate (Monat 9 und Monat 10).
3. Was ist der Unterschied zwischen der Prognose aus der letzten Übung und dem Verfahren von Holt?
4. Würden Sie denken, dass das Verfahren von Holt und Winters besser geeignet wäre, um die Verkaufszahlen zu prognostizieren?

Lösung (Python-Code):

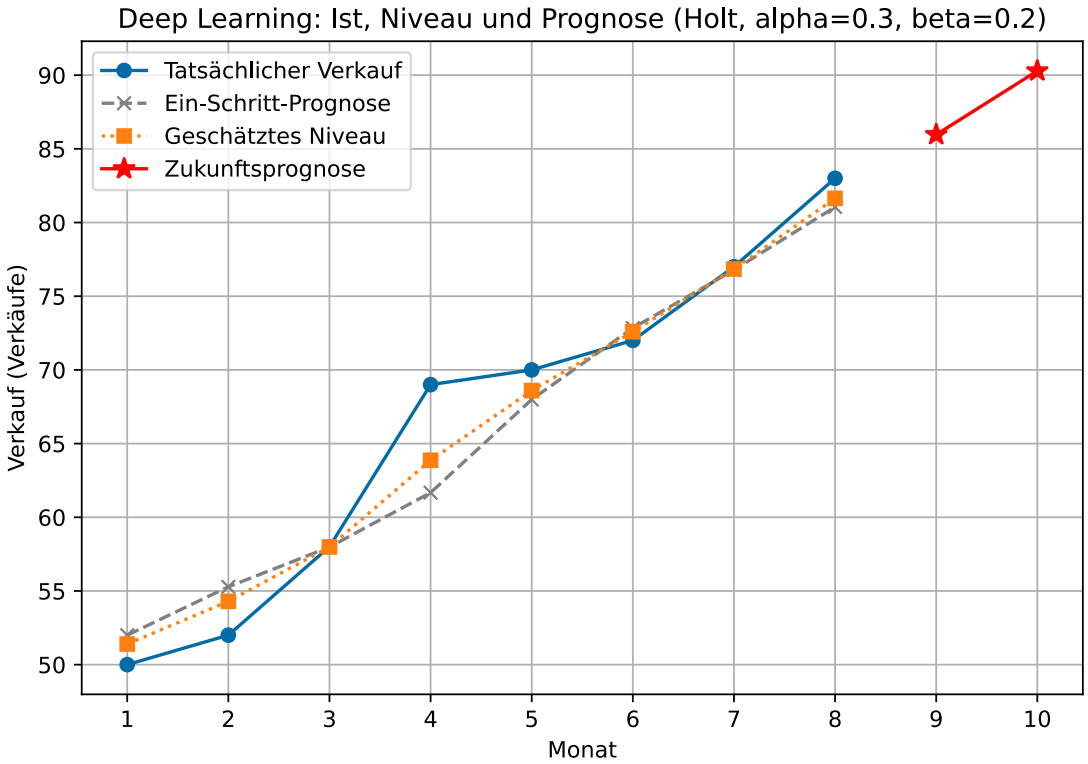
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

plt.rcdefaults()
plt.style.use('tableau-colorblind10')

# Daten für Aufgabe 2
data_task2 = {
    'Monat': list(range(1, 9)),
    'Verkauf': [50, 52, 58, 69, 70, 72, 77, 83]
}
df_task2 = pd.DataFrame(data_task2)

print("Verkaufsdaten Deep Learning:")
print(df_task2)

# Parameter für Holt's Methode
alpha_t2 = 0.3
beta_t2 = 0.2
a0_hat_t2 = 48
b0_hat_t2 = 4

# Arrays für die Speicherung der Werte
```

```
a_hat_vals_t2 = [0.0] * (len(df_task2) + 1)
b_hat_vals_t2 = [0.0] * (len(df_task2) + 1)
p_vals_t2 = [0.0] * (len(df_task2) + 1) # Prognose p_t für y_t

a_hat_vals_t2[0] = a0_hat_t2
b_hat_vals_t2[0] = b0_hat_t2

results_t2 = []
print("\nBerechnung mit Holt's Methode (t=1 bis 8):")

for t_idx in range(1, len(df_task2) + 1): # t_idx von 1 bis 8
    y_t = df_task2.loc[t_idx-1, 'Verkauf']

    # Prognose für die aktuelle Periode t (basierend auf t-1 Werten)
    # p_t = a_hat_t-1 + b_hat_t-1 (für i=1)
    p_vals_t2[t_idx] = a_hat_vals_t2[t_idx-1] + b_hat_vals_t2[t_idx-1]

    # Update Niveau
    a_hat_vals_t2[t_idx] = alpha_t2 * y_t + (1 - alpha_t2) *
(a_hat_vals_t2[t_idx-1] + b_hat_vals_t2[t_idx-1])

    # Update Trend
    b_hat_vals_t2[t_idx] = beta_t2 * (a_hat_vals_t2[t_idx] -
a_hat_vals_t2[t_idx-1]) + (1 - beta_t2) * b_hat_vals_t2[t_idx-1]

    results_t2.append({
        't': t_idx,
        'y_t': y_t,
        'p_t': p_vals_t2[t_idx], # p_t = a_hat_t-1 + b_hat_t-1
        'a_hat_t': a_hat_vals_t2[t_idx],
        'b_hat_t': b_hat_vals_t2[t_idx]
    })

df_results_t2 = pd.DataFrame(results_t2)
print(df_results_t2.round(2))

# 2. Prognose für Monat 9 und 10
# Basierend auf a_hat_8 und b_hat_8
a_hat_final_t2 = a_hat_vals_t2[len(df_task2)] # a_hat_8
b_hat_final_t2 = b_hat_vals_t2[len(df_task2)] # b_hat_8

print(f"\nLetzte Parameter (t=8):")
print(f"a_hat_8 = {a_hat_final_t2:.2f}")
print(f"b_hat_8 = {b_hat_final_t2:.2f}")

prognosen_t2_future = []
print("\nPrognosen für die nächsten 2 Monate:")
for i in range(1, 3): # i = 1 (Monat 9), i = 2 (Monat 10)
    p_future = a_hat_final_t2 + b_hat_final_t2 * i
    prognosen_t2_future.append({
        'Monat': len(df_task2) + i,
        'Prognose': p_future
```

```
    })
    print(f"Monat {len(df_task2) + i}: {p_future:.2f} Verkäufe")

df_prognosen_t2_future = pd.DataFrame(prognosen_t2_future)

# Plot für Aufgabe 2
plt.figure(figsize=(7, 5))
plt.plot(df_task2['Monat'], df_task2['Verkauf'], marker='o', linestyle='-',
label='Tatsächlicher Verkauf')
plt.plot(df_results_t2['t'], df_results_t2['p_t'], marker='x',
linestyle='--', color='gray', label='Ein-Schritt-Prognose')
plt.plot(df_results_t2['t'], df_results_t2['a_hat_t'], marker='s',
linestyle=':', label='Geschätztes Niveau')
plt.plot(df_prognosen_t2_future['Monat'],
df_prognosen_t2_future['Prognose'], marker='*', linestyle='-', color='red',
markersize=10, label='Zukunftsprognose')

plt.title(f'Deep Learning: Ist, Niveau und Prognose (Holt,
alpha={alpha_t2}, beta={beta_t2})')
plt.xlabel('Monat')
plt.ylabel('Verkauf (Verkäufe)')
all_months = list(df_task2['Monat']) +
list(df_prognosen_t2_future['Monat'])
plt.xticks(all_months)
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()
```

```
Verkaufsdaten Deep Learning:
   Monat  Verkauf
0      1       50
1      2       52
2      3       58
3      4       69
4      5       70
5      6       72
6      7       77
7      8       83

Berechnung mit Holt's Methode (t=1 bis 8):
   t  y_t    p_t  a_hat_t  b_hat_t
0  1   50  52.00    51.40     3.88
1  2   52  55.28    54.30     3.68
2  3   58  57.98    57.99     3.68
3  4   69  61.67    63.87     4.12
4  5   70  67.99    68.60     4.24
5  6   72  72.84    72.59     4.19
6  7   77  76.78    76.85     4.21
7  8   83  81.05    81.64     4.32
```

```
Letzte Parameter (t=8):
a_hat_8 = 81.64
b_hat_8 = 4.32

Prognosen für die nächsten 2 Monate:
Monat 9: 85.96 Verkäufe
Monat 10: 90.29 Verkäufe
```



3. Der Hauptunterschied liegt in der Art und Weise, wie das Niveau und der Trend der Zeitreihe geschätzt und geglättet werden. Das Verfahren aus Übung 1 (Exponentielle Glättung mit Trendkorrektur)verwendet einen einzigen Glättungsfaktor. Das Niveau und der Trend werden durch eine doppelte exponentielle Glättung derselben Zeitreihe abgeleitet. Aus diesen beiden Reihen werden dann die Schätzungen für Niveau und Trend berechnet. Die Reaktion auf Änderungen im Niveau und im Trend ist somit durch denselben Parameter alpha gekoppelt. Das Verfahren von Holt verwendet zwei separate Glättungsfaktoren: alpha für das Niveau und beta für den Trend. Das Niveau wird direkt aus dem aktuellen Beobachtungswert y_t und der vorherigen Schätzung für Niveau und Trend (a_t-1 + b_t-1) berechnet. Der Trend (b_t) wird direkt aus der Veränderung des Niveaus (a_t - a_t-1) und der vorherigen Trendschätzung (b_t-1) berechnet. Dies ermöglicht eine flexiblere Anpassung, da man unabhängig voneinander festlegen kann, wie stark die Glättung auf Veränderungen im Niveau bzw. im Trend reagieren soll. Wenn sich beispielsweise das Grundniveau schnell ändert, der Trend aber relativ stabil ist, kann man unterschiedliche Werte für alpha und beta wählen. Zusammenfassend lässt sich sagen, dass das Verfahren von Holt durch die Verwendung von zwei getrennten Glättungsfaktoren eine differenziert-

ere Anpassung an Niveau und Trend einer Zeitreihe erlaubt als die exponentielle Glättung mit Trendkorrektur, die nur einen Glättungsfaktor nutzt.

4. Das Verfahren von Holt-Winters erweitert das Holt-Verfahren um eine saisonale Komponente. Es ist also besonders dann geeignet, wenn die Zeitreihe nicht nur einen Trend, sondern auch wiederkehrende saisonale Muster aufweist (z.B. quartalsweise oder monatliche Schwankungen, die sich jedes Jahr ähnlich wiederholen). Diese Daten zeigen einen deutlichen Trend (steigende Verkaufszahlen). Ob auch eine signifikante Saisonalität vorliegt, lässt sich anhand von nur 8 Monaten Daten schwer beurteilen. Für die Identifizierung von saisonalen Mustern benötigt man in der Regel Daten über mehrere Saisons hinweg (z.B. mindestens 2-3 Jahre bei monatlichen Daten, um zu sehen, ob sich bestimmte Monate systematisch von anderen unterscheiden). Mit den aktuell vorliegenden 8 Datenpunkten ist das Verfahren von Holt eine gute Wahl, da es den sichtbaren Trend adressiert. Um zu entscheiden, ob Holt-Winters besser wäre, müssten mehr Daten gesammelt werden. Wenn sich dann klare saisonale Muster zeigen, wäre Holt-Winters eine sinnvolle Erweiterung. Andernfalls ist Holt ausreichend.