Bakery Sales Prediction

Team “Room 2”

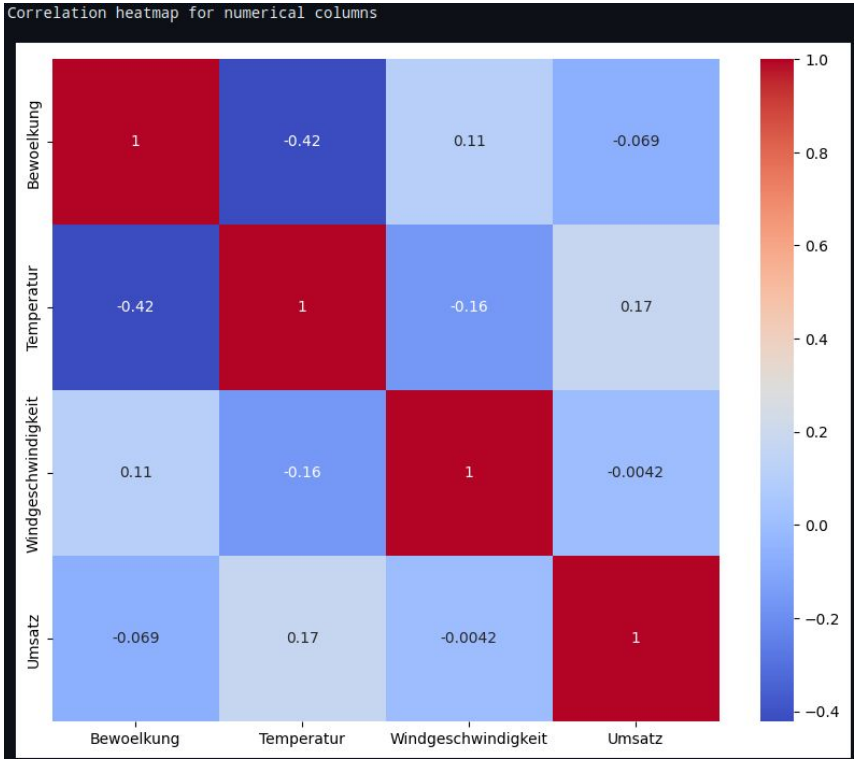
Sascha Mahmood
Kacper Nyka
David Schubert

Agenda

- Variablen - Sascha
- Konfidenzintervalle - Sascha
- Optimierung Lineares Modell - Kacper
- Missing Value Imputation - Sascha
- Neuronales Netz - Sascha
- Worst Fail / Best Improvement - David

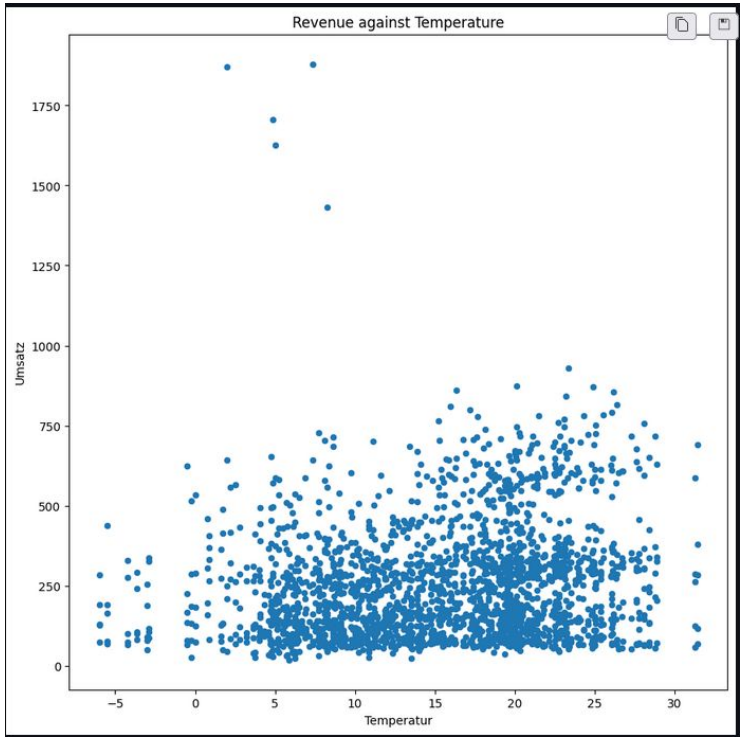
Variablen

Temperatur	Warengruppe_2	Warengruppe_3	Warengruppe_4	Warengruppe_5	Warengruppe_6	summer	autumn	StartOfMonth



Variablen Temperatur

Temperatur	Warengruppe_2	Warengruppe_3	Warengruppe_4	Warengruppe_5	Warengruppe_6	summer	autumn	StartOfMonth



Variablen Sommerferien

Temperatur	Warengruppe_2	Warengruppe_3	Warengruppe_4	Warengruppe_5	Warengruppe_6	summer	autumn	StartOfMonth

```
sommer_stats = df.groupby("summer")["Umsatz"].describe()
autumn_stats = df.groupby("autumn")["Umsatz"].describe()
som_stats = df.groupby("StartOfMonth")["Umsatz"].describe()
```

```
          count        mean         std        min         25%         50%  \
summer
0        1078.0  213.887366  169.528471  18.483993  100.870030  166.415130
1        1106.0  291.055932  187.592710  36.344812  137.241356  270.581136

                75%          max
summer
0        284.085577  1879.461831
1        372.196978   930.801703
```

Variablen Herbstferien

Temperatur Warengruppe_2 Warengruppe_3 Warengruppe_4 Warengruppe_5 Warengruppe_6 summer autumn StartOfMonth

```
sommer_stats = df.groupby("summer")["Umsatz"].describe()
autumn_stats = df.groupby("autumn")["Umsatz"].describe()
som_stats = df.groupby("StartOfMonth")["Umsatz"].describe()
```

autumn	count	mean	std	min	25%	50%	\
0	1858.0	261.585410	190.517597	18.483993	116.226568	222.681159	
1	326.0	203.842795	121.042849	23.097747	105.223297	161.846654	

autumn	75%	max
0	337.967097	1879.461831
1	286.661081	704.513358

Variablen Anfang des Monats

Temperatur	Warengruppe_2	Warengruppe_3	Warengruppe_4	Warengruppe_5	Warengruppe_6	summer	autumn	StartOfMonth

```
sommer_stats = df.groupby("summer")["Umsatz"].describe()
autumn_stats = df.groupby("autumn")["Umsatz"].describe()
som_stats = df.groupby("StartOfMonth")["Umsatz"].describe()
```

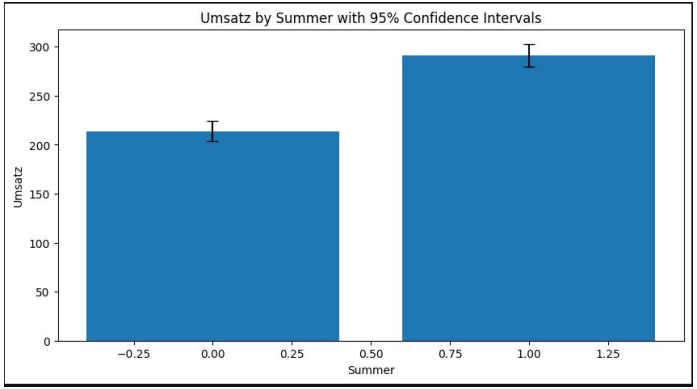
StartOfMonth	count	mean	std	min	25%	\
0	1765.0	255.574847	186.053668	23.097747	115.775782	
1	419.0	241.978131	169.185213	18.483993	102.953506	

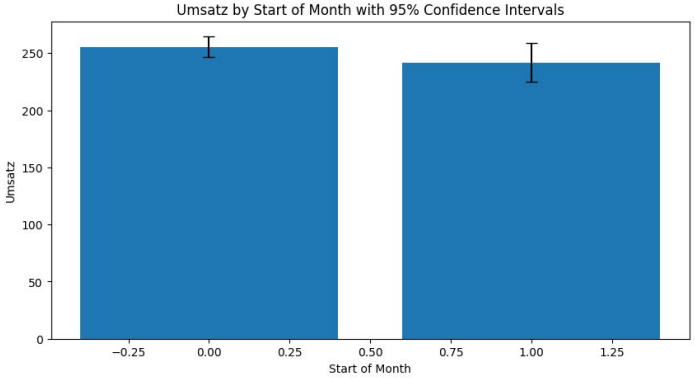
StartOfMonth	50%	75%	max
0	215.929339	332.713957	1879.461831
1	191.050544	315.201717	930.801703

Konfidenzintervalle Sommerferien und Monatsanfang

Temperatur Warengruppe_2 Warengruppe_3 Warengruppe_4 Warengruppe_5 Warengruppe_6 summer autumn StartOfMonth

-





```
mod = smf.ols('Umsatz ~ summer * autumn * StartOfMonth * C(Warengruppe)', data=df).fit()
```

Lineares Modell

OLS Regression Results

Dep. Variable:	Umsatz	R-squared:	0.737
Model:	OLS	Adj. R-squared:	0.733
Method:	Least Squares	F-statistic:	188.1
Date:	Mon, 24 Jun 2024	Prob (F-statistic):	0.00
Time:	21:45:08	Log-Likelihood:	-13018.
No. Observations:	2184	AIC:	2.610e+04
Df Residuals:	2151	BIC:	2.629e+04
Df Model:	32		
Covariance Type:	nonrobust		

Modellübersicht:

- R-squared: 0.737 (73,7% der Varianz in der abhängigen Variable wird durch das Modell erklärt)
- Adjusted R-squared: 0.733
- F-statistic: 188.1 mit einem p-Wert von 0.00, was auf eine hohe statistische Signifikanz des Gesamtmodells hinweist

Zusammenfassend

- Das Modell erklärt einen erheblichen Teil der Varianz im Umsatz, und viele der Haupt- und Wechselwirkungseffekte sind signifikant.
- Allerdings gibt es Hinweise auf **Multikollinearität** und eine starke Abweichung der Residuen von der Normalverteilung, was die Ergebnisse beeinflussen könnte.

```
('log_Umsatz ~ summer + autumn + StartOfMonth + C(Warengruppe) + summer:C(Warengruppe) + autumn:C(Warengruppe)',
```

Optimierung des lineares Modell

OLS Regression Results

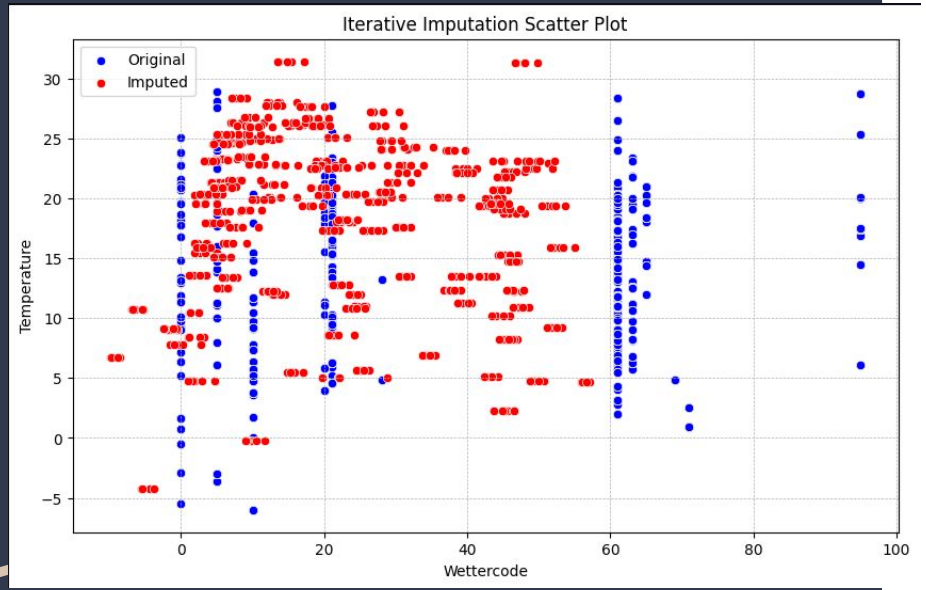
Dep. Variable:	log_Umsatz	R-squared:	0.834
Model:	OLS	Adj. R-squared:	0.832
Method:	Least Squares	F-statistic:	638.4
Date:	Mon, 24 Jun 2024	Prob (F-statistic):	0.00
Time:	21:55:17	Log-Likelihood:	-404.47
No. Observations:	2184	AIC:	844.9
Df Residuals:	2166	BIC:	947.3
Df Model:	17		
Covariance Type:	nonrobust		

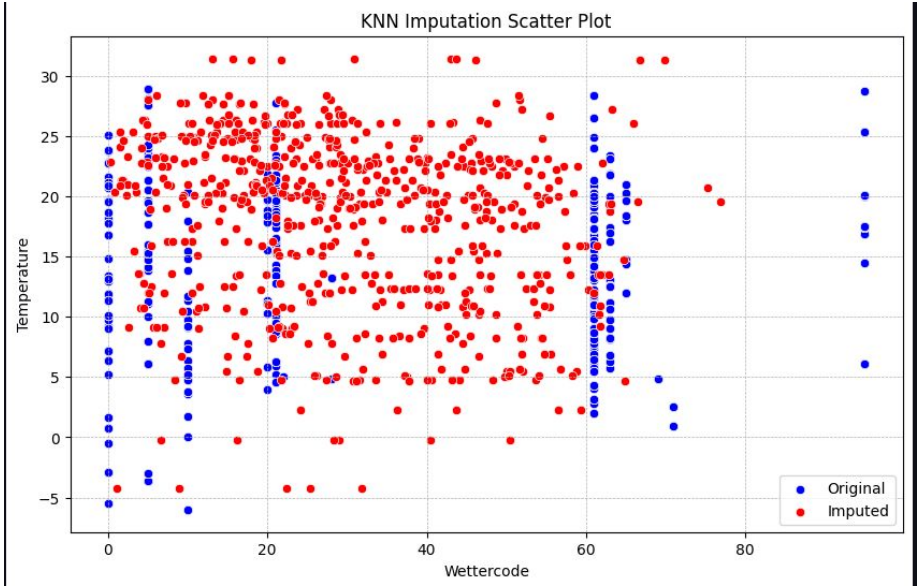
Modellübersicht:

- **R-squared:** 0.834 - Das Modell erklärt nun 83,4% der Varianz in log_Umsatz, was eine erhebliche Verbesserung darstellt.
- **Adjusted R-squared:** 0.832 - Bestätigt die Verbesserung unter Berücksichtigung der Anzahl der Prädiktoren.
- **F-statistic:** 638.4, was auf eine hohe Gesamtbedeutung des Modells hinweist ($p < 0.001$).

- **logarithmierter Umsatz** als abhängige Variable scheint die Modellanpassung verbessert zu haben.
- Die Erklärungskraft ist gestiegen, die Residuen sind näher an einer Normalverteilung, und mehr abh. Variablen sind signifikant (wie summer und autumn).
- Multikollinearität bleibt aber immer noch ein Problem.

Missing Value Imputation





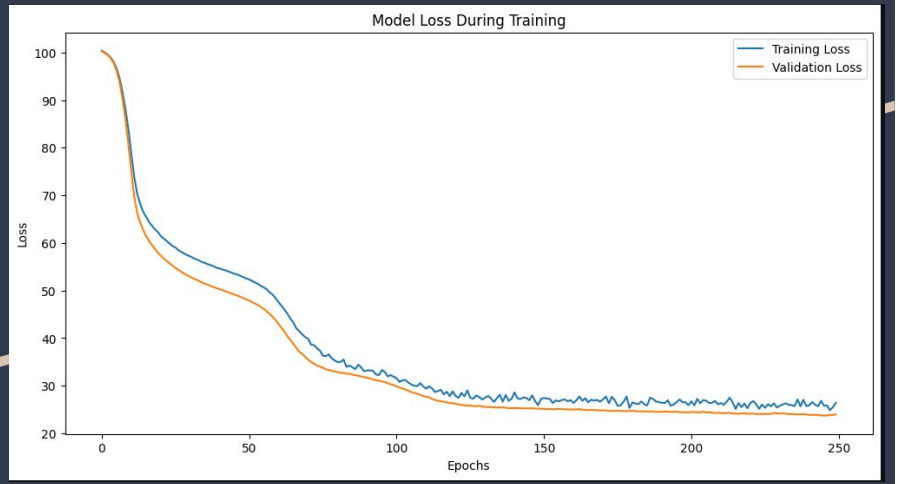
Neuronales Netz

```
model.compile(loss="mape", optimizer=Adam(learning_rate=0.001))

history = model.fit(
    training_features,
    training_labels,
    epochs=250,
    validation_data=(validation_features, validation_labels)
)
```

Neuronales Netz DNN

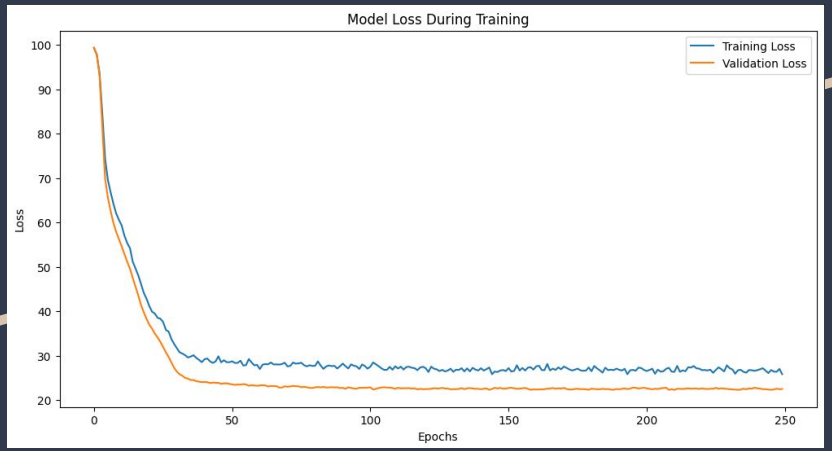
Layer (type)	Output Shape	Param #
batch_normalization_13 (BatchNormalization)	(None, 9)	36
dense_44 (Dense)	(None, 10)	100
dense_45 (Dense)	(None, 1)	11




```
loss: 27.1396 - val_loss: 25.6048
```

Neuronales Netz DNN

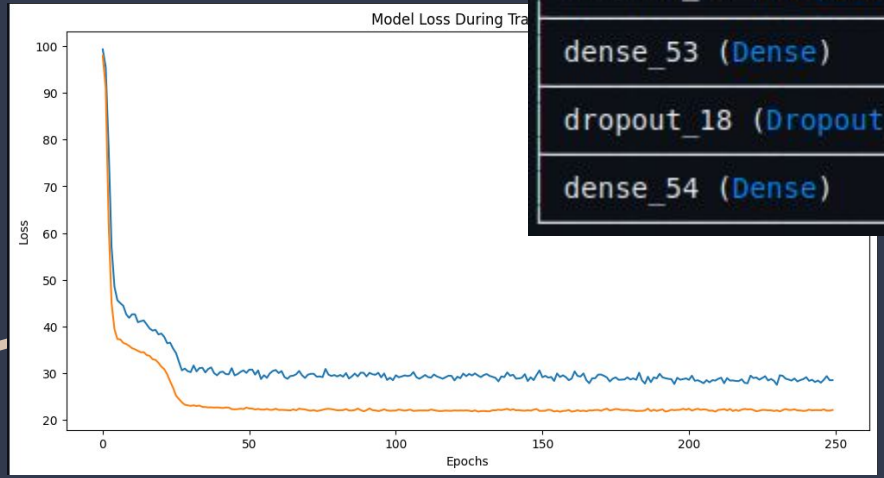
Layer (type)	Output Shape	Param #
batch_normalization_14 (BatchNormalization)	(None, 9)	36
dense_46 (Dense)	(None, 10)	100
dense_47 (Dense)	(None, 20)	220
dropout_16 (Dropout)	(None, 20)	0
dense_48 (Dense)	(None, 1)	21




```
loss: 25.3988 - val_loss: 22.5239
```

Neuronales Netz DNN

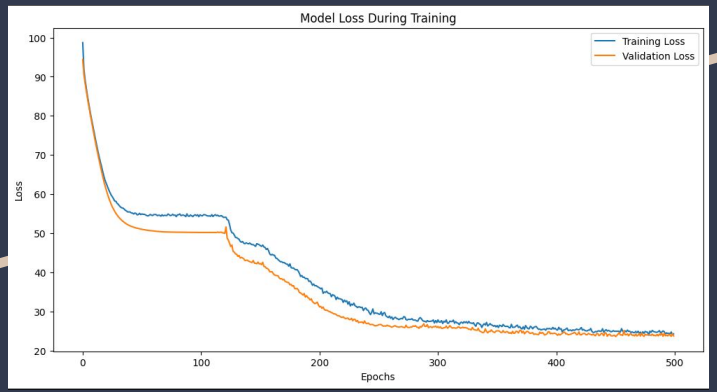
Layer (type)	Output Shape	Param #
batch_normalization_16 (BatchNormalization)	(None, 9)	36
dense_51 (Dense)	(None, 10)	100
dense_52 (Dense)	(None, 20)	220
dropout_17 (Dropout)	(None, 20)	0
dense_53 (Dense)	(None, 20)	420
dropout_18 (Dropout)	(None, 20)	0
dense_54 (Dense)	(None, 1)	21




```
loss: 27.6795 - val_loss: 22.1171
```

Neuronales Netz LSTM

Layer (type)	Output Shape	Param #
lstm (LSTM)	(None, 50)	10,400
dropout_2 (Dropout)	(None, 50)	0
dense_4 (Dense)	(None, 1)	51



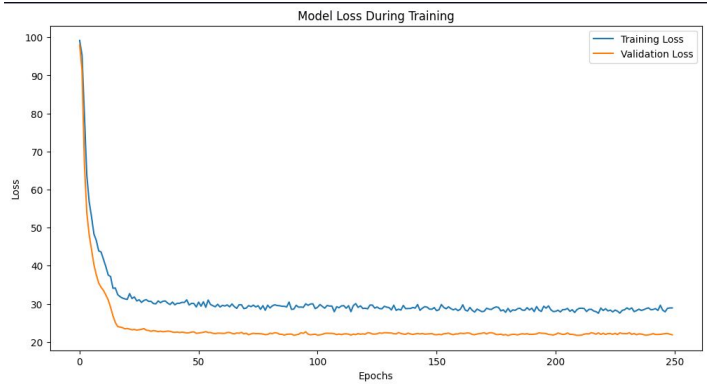

```
loss: 24.0814 - val_loss: 23.7407
```

Neuronales Netz MAPE

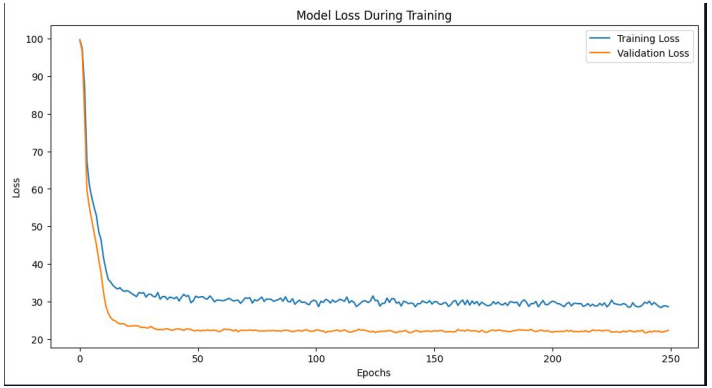
```
MAPE:
##############
Complete:
0.2197135092107639
Warengruppe_1
0.281752520443913 97
Warengruppe_2
0.1740325292019712
Warengruppe_3
0.247952477063705 05
Warengruppe_4
0.2231292266943377
Warengruppe_5
0.163286918053793 72
Warengruppe_6
0.421646499517111 53
```

Neuronales Netz MAPE

Mit Temperatur



Ohne Temperatur



Worst Fail

Probleme:

- Multikollinearität
- Overfitting

```
import statsmodels.formula.api as smf
mod = smf.ols('Umsatz ~ summer * autumn * StartOfMonth * C(Warengruppe)', data=df).fit()
print(mod.summary())
```

- R-squared: 0.737
- Viele Wechselwirkungen nicht signifikant

.

Beispielhafte Koeffizienten:

- `summer:C(Warengruppe)[T.2]`: 166.0978 ($p < 0.0001$)
- `summer:C(Warengruppe)[T.6]`: 7.413e-14 ($p = 0.665$)

Best Improvement

Ansatz:

- Vereinfachung des Modells
- Reduktion Interaktionsanzahl
- Fokus auf Hauptvariabeln

Vorher

summer:C(Warengruppe)[T.2]: 166.0978 (p < 0.0001)

summer:C(Warengruppe)[T.6]: 7.413e-14 (p = 0.665)

```
mod = smf.ols('Umsatz ~ summer + autumn + StartOfMonth + C(Warengruppe) + summer:C(Warengruppe) + autumn:C(Warengruppe)', data=df).fit()
print(mod.summary())
```

R-squared: 0.733

Verbesserte Signifikanz der Koeffizienten.

Beispielhafte Koeffizienten:

Angepasst

summer:C(Warengruppe)[T.2]: 165.9782 (p < 0.0001)

summer:C(Warengruppe)[T.6]: 3.503e-14 (p = 0.163)

Vorteile

- **Reduzierte Multikollinearität:**
  - Entfernen unnötiger Wechselwirkungen
- **Bessere Generalisierung:**
  - Das vereinfachte Modell zeigte bessere Ergebnisse bei Validierungsdaten