

Project Presentation - 08.01.2026

Sales Forecasting for a Bakery Branch

Introduction to Data Science and Machine Learning - Group 11

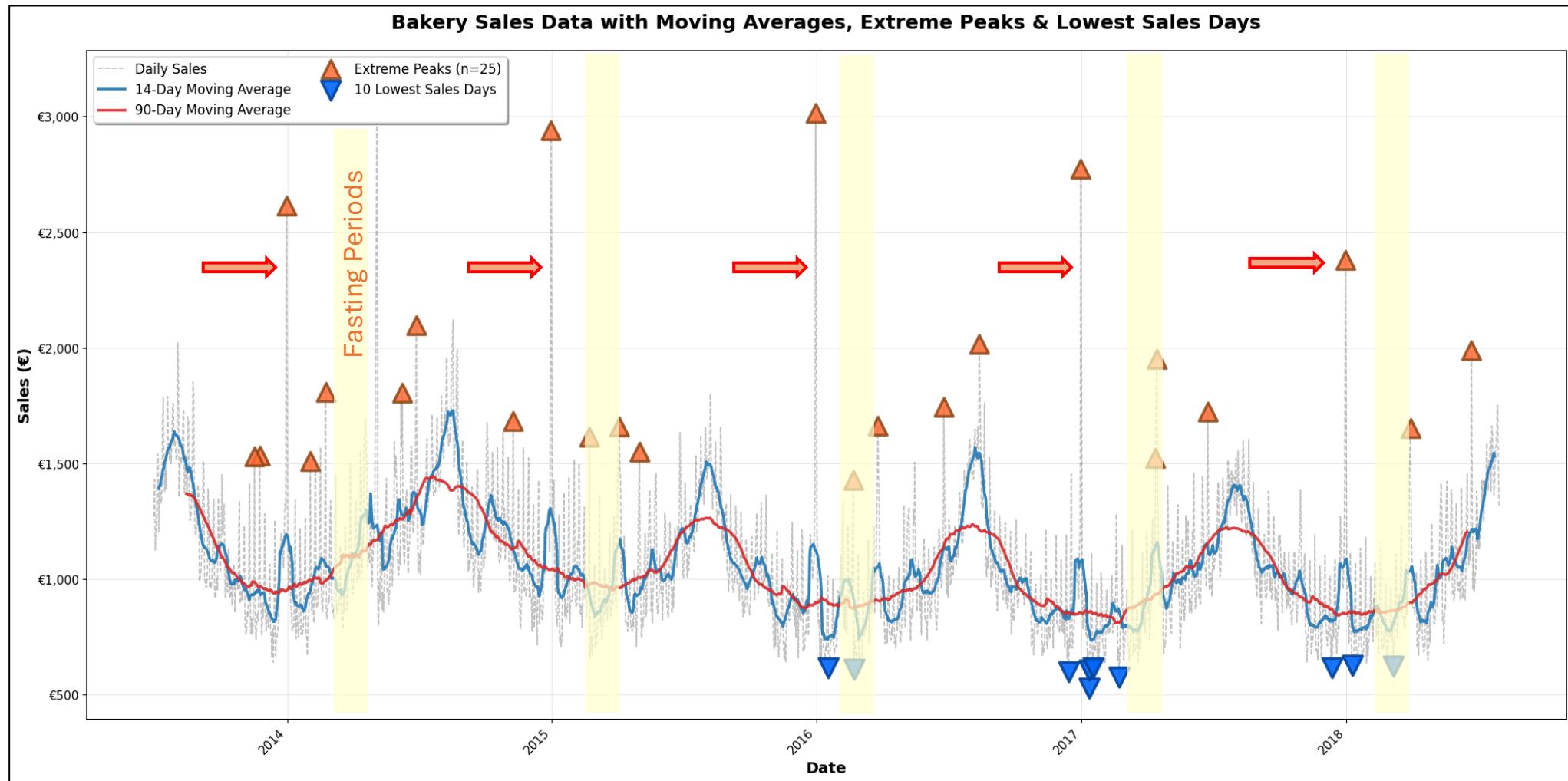
Clara Brilke

Lucia Linder (Matr.-Nr.: 948064)

Lasse Ludek (Matr.-Nr.: 1159543)

Johanna Forker (Matr.-Nr.: 1158656)

Revenue Trends and Extremes (2013–2019)



Self created variables

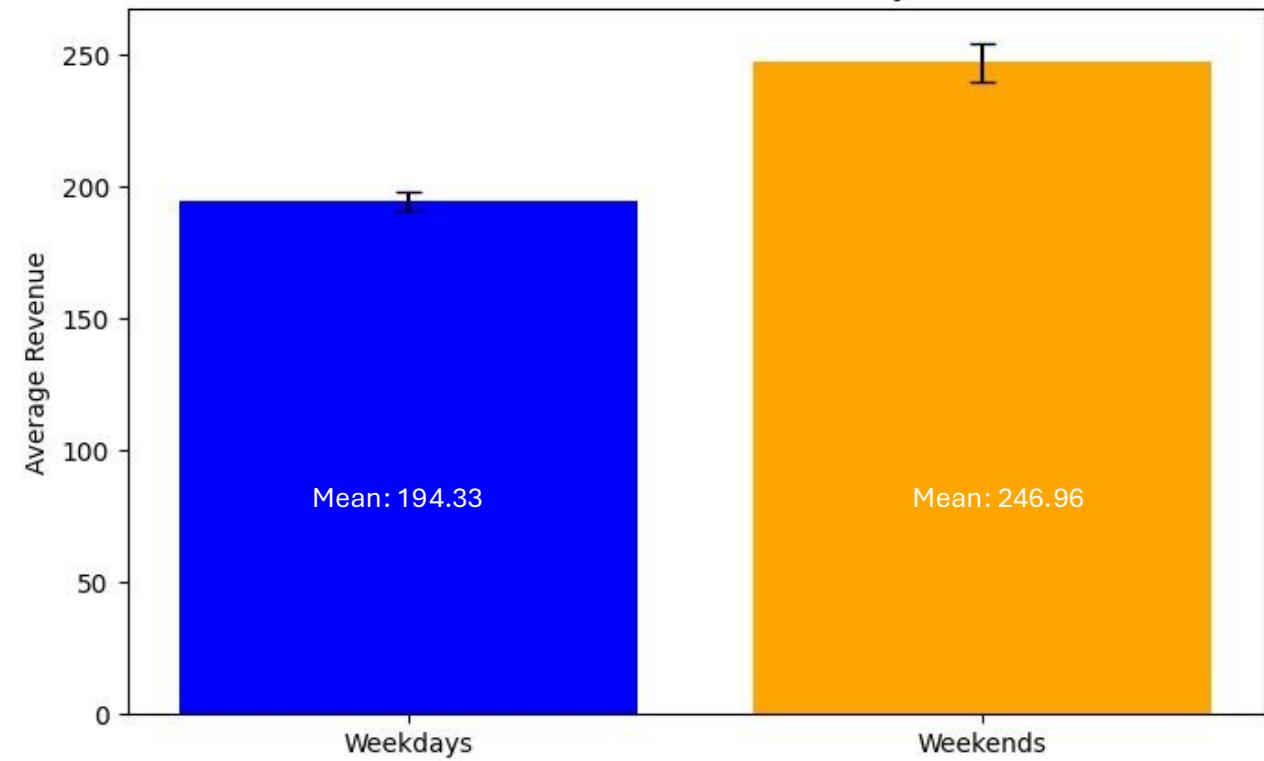
- **German Holidays**
 - Contains dates for German holidays
 - Each date is labeled with a binary indicator
 - 1 = german public holiday
 - 0 = regular working day
- **Pre-Holiday-Indicator**
 - Based on German holidays, derived a list of pre-holiday dates
 - Each date is labeled with a binary indicator
 - 1 = day immediately preceding a public holiday
 - 0 = regular day that does not precede a public holiday
- **Fasting Periods**
 - Contains calendar dates for the Christian fasting period from 2013–2019
 - Each date is labeled with a binary indicator
 - 1 = date within the fasting period
 - 0 = date outside the fasting period
- **Weekdays**
 - Contains calendar dates for weekdays
 - Derived from given Sales Data
 - Each date is labeled with a binary indicator
 - 1 = weekday
 - 0 = weekend

Self created variables

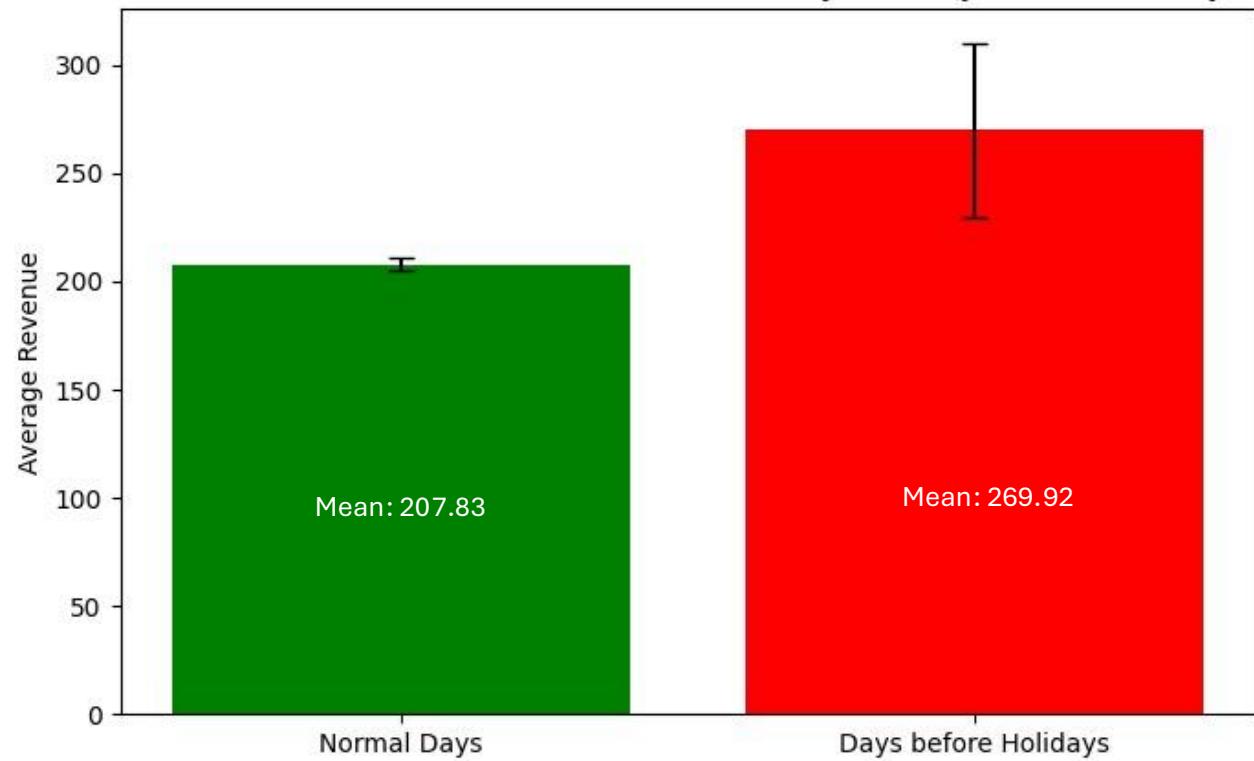
- **Months**
 - Contains calendar months
 - Derived from given Sales Data
 - Each month is labeled with a binary indicator
 - 1 = German public holiday on that date
 - 0 = regular working day (no public holiday)
- **LAG-Data**
 - Lag_1: Represents the sales value from the previous day.
 - Captures short-term temporal dependencies and day-to-day dynamics in the data.
 - Lag_7: Represents the sales value from the same day one week earlier.
 - Models weekly patterns and recurring behaviors in daily sales data.
 - Rolling_7: Represents the 7-day rolling average of sales.
 - Smooths short-term fluctuations and provides a stable estimate of recent sales trends.
- **‘Kieler Woche’**
 - Contains calendar dates related to the Kieler Woche event
 - Each date is labeled with a binary indicator
 - 1 indicates that Kieler Woche takes place on that date
 - 0 indicates that Kieler Woche does not take place on that date
- **Weather Data**
 - Contains daily weather observations
 - Includes the following variables: Date, Cloud Cover, Temperature, Wind Speed, Weather Code

Confidence Intervals

Confidence Intervals: Revenue on Weekdays vs. Weekends



Confidence Intervals: Revenue on Normal Days vs. Days before Holidays



Missing Value Imputation

- **Missing temperature values** were imputed using official data from the German Weather Service (DWD)
 - Temperature data was taken from the Kiel Holtenau climate station
 - Missing temperature columns were filled with the corresponding values from the DWD dataset
- **Missing weather code values** were grouped into a separate residual category
 - We deliberately avoided imputing weather codes based on neighboring days due to high uncertainty and large gaps in the data

Imputation of Missing Temperature Values

```
# Ergänze fehlende Temperaturdaten aus DWD-Daten
df_wetter_dwd = pd.read_csv('data/df_wetterdaten_dwd.csv')
df_wetter_dwd['Datum'] = pd.to_datetime(df_wetter_dwd['Datum'], errors='coerce')

# Ergänze fehlende Angaben der Temperatur aus DWD-Daten

# Merge über gemeinsame Spalte (z.B. 'Datum')
df_merged = df_merged.merge(df_wetter_dwd[['Datum', 'Temperatur']],
                            on='Datum',
                            how='left',
                            suffixes=('', 'DWD'))

# Fehlende Werte ergänzen
df_merged['Temperatur'] = df_merged['Temperatur'].fillna(df_merged['TemperaturDWD'])

print ('Fehlende Temperaturwerte:')
print(df_merged['Temperatur'].isna().sum()) # Überprüfe, ob noch fehlende Werte vorhanden sind

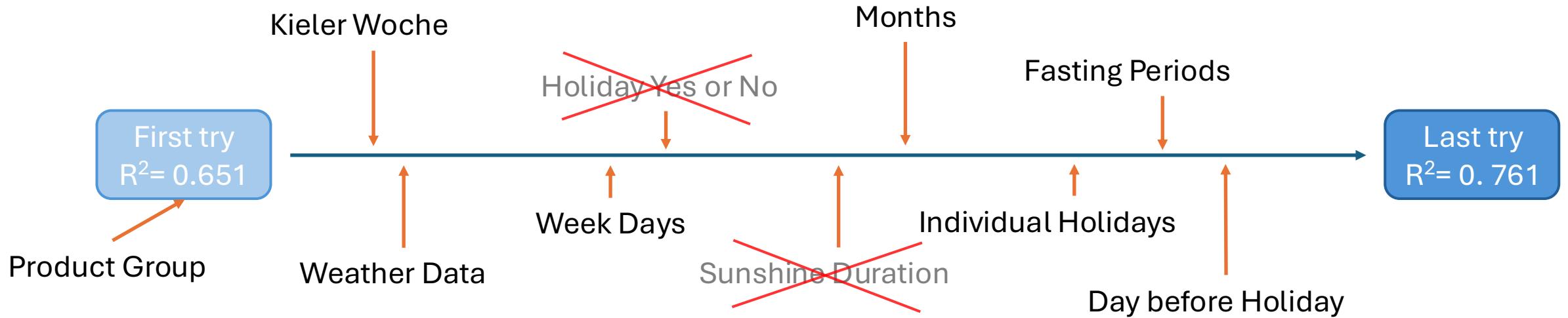
print(df_merged[df_merged['Temperatur'].isna()][['Datum', 'Temperatur', 'TemperaturDWD']])
```

Weather Code Categorization and Residual NA Category

```
wettercode_mapping = {  
    0.0: 'Klar',  
    5.0: 'Dunst',  
    10.0: 'Dunst',  
    17.0: 'Extremwetter',  
    20.0: 'NachSchlechtemWetter',  
    21.0: 'NachSchlechtemWetter',  
    22.0: 'NachSchlechtemWetter',  
    28.0: 'NachSchlechtemWetter',  
    45.0: 'Nebel',  
    49.0: 'Nebel',  
    53.0: 'Regen',  
    55.0: 'Regen',  
    61.0: 'Regen',  
    63.0: 'Regen',  
    65.0: 'Regen',  
    68.0: 'Regen',  
    69.0: 'Regen',  
    71.0: 'Schnee',  
    73.0: 'Schnee',  
    75.0: 'Schnee',  
    77.0: 'Schnee',  
    79.0: 'Extremwetter',  
    95.0: 'Extremwetter',  
    'KeineDaten': 'KeineDaten'  
}
```

Linear Model Optimization

- General Procedure: First Creating different variables, after model training looking into the results at $P>|t|$
- Saw that the model was overfitting: we had created multiple weather variables which were very similar, took them out
- Variables that improved our model significantly: Feiertage, TagVorFeiertag, Wochentag, Monat



Linear Model Optimization

OLS Regression Results

Dep. Variable:	Umsatz	R-squared:	0.762
Model:	OLS	Adj. R-squared:	0.761
Method:	Least Squares	F-statistic:	643.9
Date:	Tue, 06 Jan 2026	Prob (F-statistic):	0.00
Time:	13:42:18	Log-Likelihood:	-42655.
No. Observations:	7487	AIC:	8.539e+04
Df Residuals:	7449	BIC:	8.565e+04
Df Model:	37		
Covariance Type:	nonrobust		

Neural Network Optimization

- Playing with the architecture of the neural net
- Created LAG variables Umsatz_lag_1 Umsatz_lag_7, Umsatz_rolling_7 which improved our model

```
df_merged['Umsatz_lag_1'] = df_merged['Umsatz'].shift(1) # Gestern  
df_merged['Umsatz_lag_7'] = df_merged['Umsatz'].shift(7) # Vor einer Woche  
df_merged['Umsatz_rolling_7'] = df_merged['Umsatz'].rolling(7).mean() # Durchschnitt letzte 7 Tage
```

Neural Network Optimization

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, BatchNormalization, Dropout
from tensorflow.keras.optimizers import Adam

model = Sequential([
    Dense(64, activation='relu', input_shape=(X_train_scaled.shape[1],)),
    BatchNormalization(),
    Dropout(0.2),

    Dense(32, activation='relu'),
    BatchNormalization(),
    Dropout(0.2),

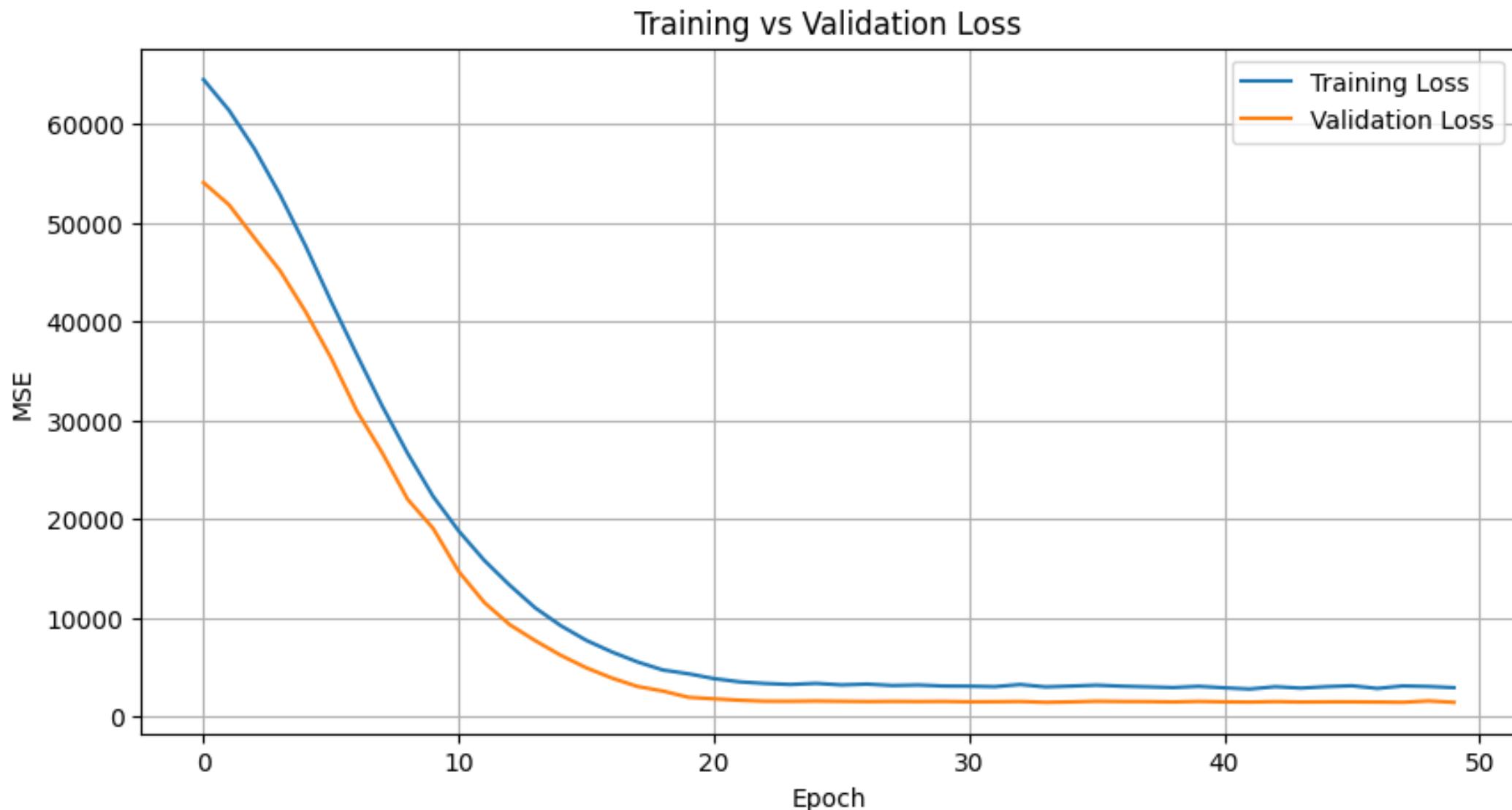
    Dense(1)
])

model.compile(
    optimizer=Adam(learning_rate=0.001),
    loss='mse'
)

model.summary()

history = model.fit(
    X_train_scaled,
    y_train,
    validation_data=(X_val_scaled, y_val),
    epochs=50,
    batch_size=64,
    verbose=1
)
```

Neural Network Optimization



Neural Network Optimization

Training:

MAE: **25.05 €** (Durchschnitt der absoluten Fehler)
RMSE: **35.66 €** (Wurzel aus mittlerem quadratischen Fehler: bestraft große Fehler stärker)
 R^2 : **0.942** (Bestimmtheitsmaß: 0-1, höher = besser)
MAPE: **16.20 %** (Durschnittlicher prozentualer Fehler)

Validation:

MAE: **27.16 €** (Durchschnitt der absoluten Fehler)
RMSE: **38.40 €** (Wurzel aus mittlerem quadratischen Fehler: bestraft große Fehler stärker)
 R^2 : **0.913** (Bestimmtheitsmaß: 0-1, höher = besser)
MAPE: **18.04 %** (Durschnittlicher prozentualer Fehler)

Formeln:

MAE: $(\sum|y_{true} - y_{pred}|)/n$
RMSE: $\sqrt{[(\sum(y_{true} - y_{pred})^2)/n]}$
 R^2 : $1 - (\sum(y_{true} - y_{pred})^2 / \sum(y_{true} - \bar{y})^2)$
MAPE: $(1/n) * \sum|(y_{true} - y_{pred})/y_{true}| * 100$

MAPE scores for each product group

WarengruppeBread: 17.03%

WarengruppeRolls: 9.91%

WarengruppeCroissants: 15.66%

WarengruppeConfectionery: 23.99%

WarengruppeCake: 13.47%

WarengruppeSeasonalBread: 46.94%

Worst Fail & Best Improvement

- **Worst Fail**
 - Initially used a single binary column: **1 = holiday, 0 = non-holiday**
 - Not reliable! Bakery **operates on some holidays** (e.g., 31.12.) and is **closed on others** (e.g., 01.01.)
 - A simple holiday indicator **did not capture actual sales patterns**
- **Best Improvement**
 - Split holidays into **individual categories** to capture the **impact of each holiday**
 - Added a **pre-holiday category** to account for customers shopping **before days the bakery is closed**