Einführung in Data Science und maschinelles Lernen

Projektpräsentation - Gruppe 10

Michael Walla, Vitali Sorin, Pierre Mayer und Aaron Schmitt

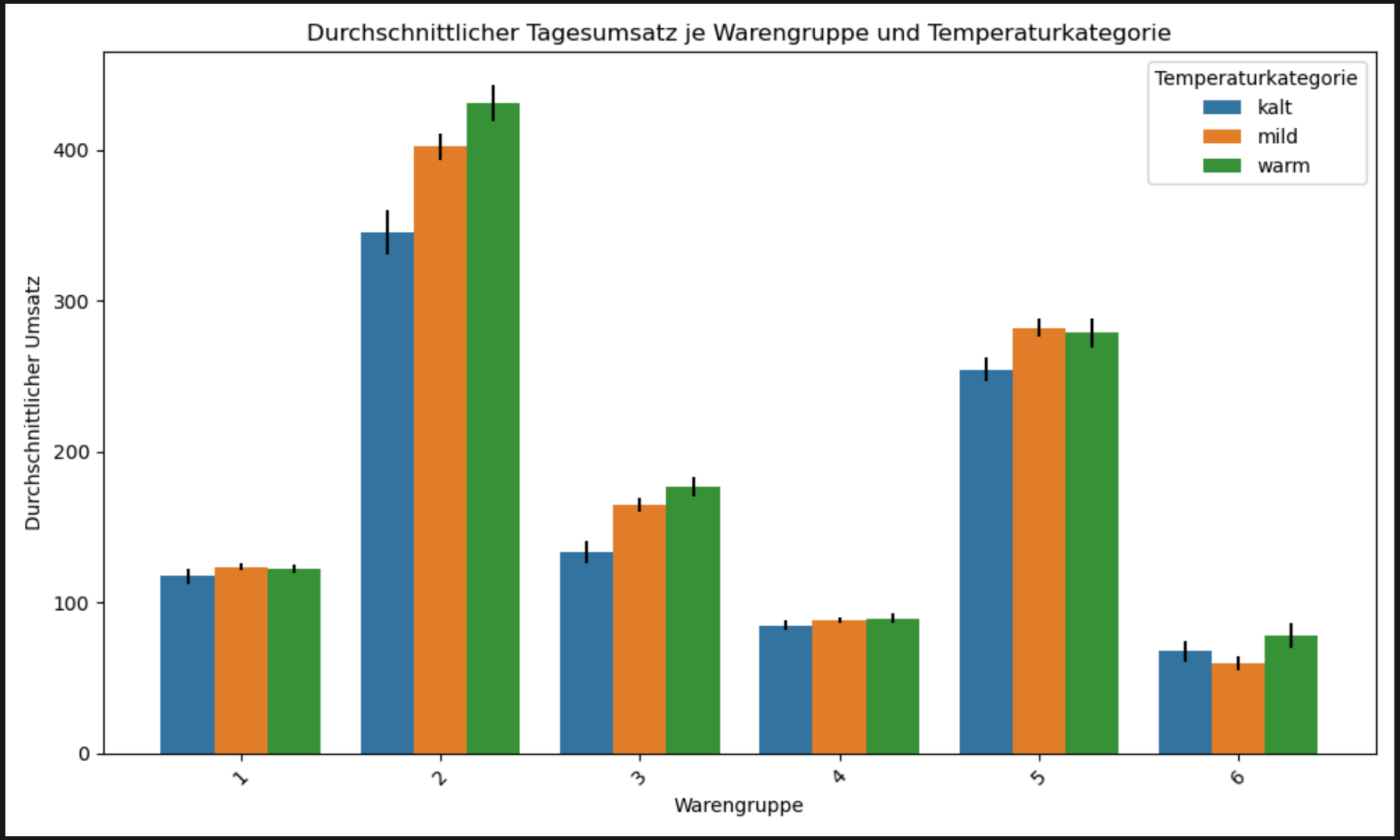
Variabeln

- Wochentag (Tag als Zahl codiert Montag = 0 bis Sonntag = 6)
- Wettercodes
  - Niederschlag (Schnee - 1, Regen - 2 und gemischt -3)
  - Intensität Niederschlag (leicht - 1, mittel - 2, stark - 3)
  - Gewitter (boolean)

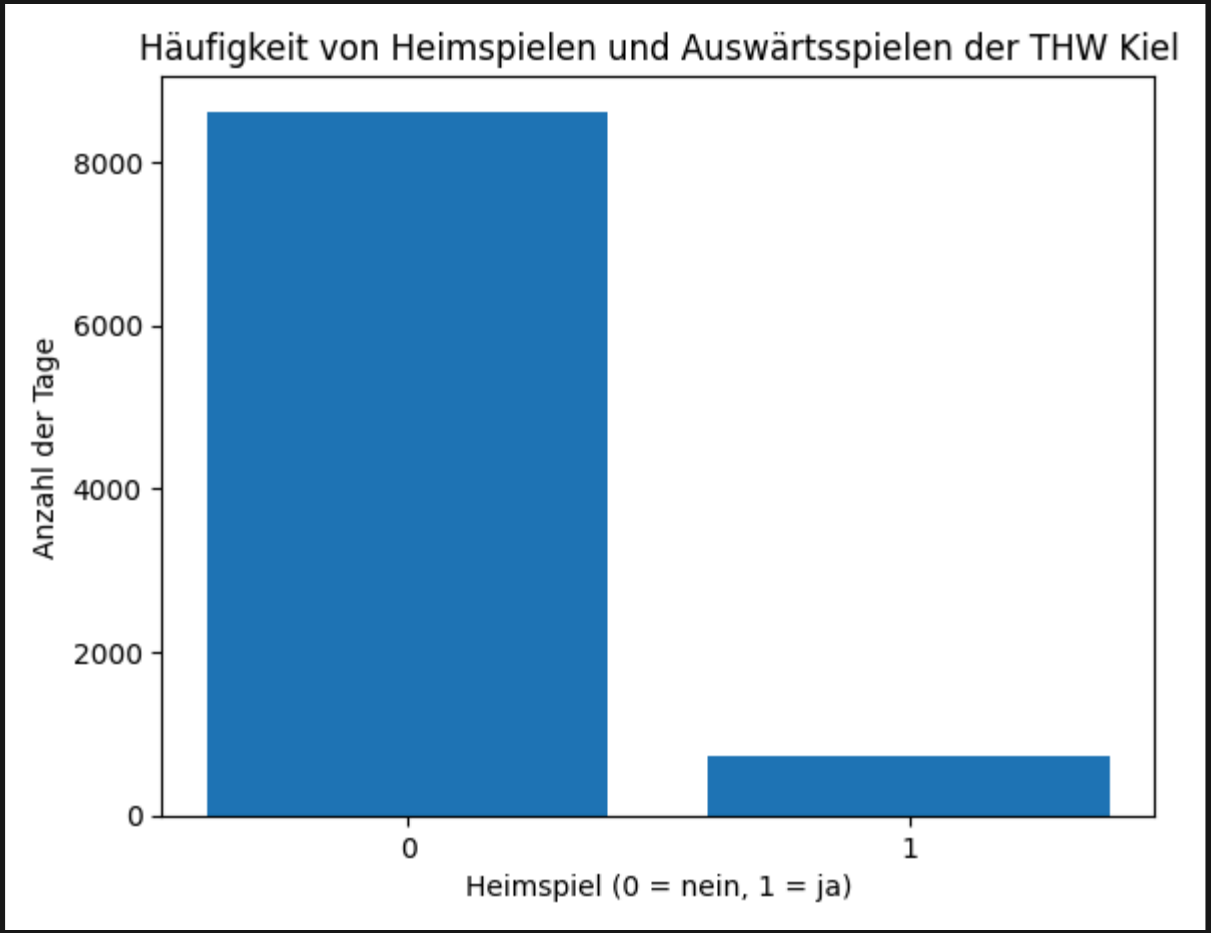
- Temperaturen eingeteilt in Bereiche je Jahreszeit
  - kalt - 0 (Frühjahr/Herbst < 8°, Sommer < 16°, Winter < -2°)
  - mild - 1 (Frühjahr/Herbst 8 - 15, Sommer 16 - 22, Winter -2 - 5)
  - warm - 2 (Frühjahr/Herbst >= 16, Sommer >= 22, Winter >= 5)

- Sportveranstaltungen
  - Heimspiel THW Kiel
  - Heimspiele Holstein Kiel
- Schulferien (boolean)
- Feiertage (boolean)

Temperaturkategorie



Heimspiele THW Kiel



Optimierung des linearen Modells!

Modellbewertung

MSE: 7629.32, R^2: 0.65

R^2 bedeutet: 65 % der Varianz des Umsatzes durch das Modell erklärt.
Bedeutet RMSE= 87.34, durchschnittliche Abweichung Vorhersagen vom tatsächlichen Umsatz

Lineare Modellgleichung:

Umsatz = 121.34 + 289.26Warengruppe_2.0 + 42.81 Warengruppe_3.0 -33.01Warengruppe_4.0 + 159.55 Warengruppe_5.0 - 54.36* Warengruppe_6.0

Missing Value Imputation

- Bei bekannten vollständigen Variabeln: .fillna(0)
- listwise deletion

Ausblick

- fehlende Wetterdaten ggf. durch Daten vom DWD ergänzen
  - Prediction basierend auf den beiden Datensätzen

Optimierung des neuronalen Netzes

Source Code

Definition

```
model = Sequential([
    # Eingabeschicht mit stärkerer Regularisierung
    Dense(64, activation='relu',
          input_dim=X_train_scaled.shape[1],
          kernel_regularizer=l2(0.01)),
    BatchNormalization(),
    Dropout(0.3),
    # Mittlere Schicht
    Dense(32, activation='relu',
          kernel_regularizer=l2(0.01)),
    BatchNormalization(),
    Dropout(0.2),
    # Letzte versteckte Schicht
    Dense(16, activation='relu',
```

Optimierer und Callbacks

```
optimizer = Adam(learning_rate=0.001)

early_stopping = EarlyStopping(
    monitor='val_loss',
    patience=20,
    restore_best_weights=True,
    mode='min')

checkpoint = ModelCheckpoint(
    'best_model.keras',
    monitor='val_loss',
    save_best_only=True,
    mode='min')
```

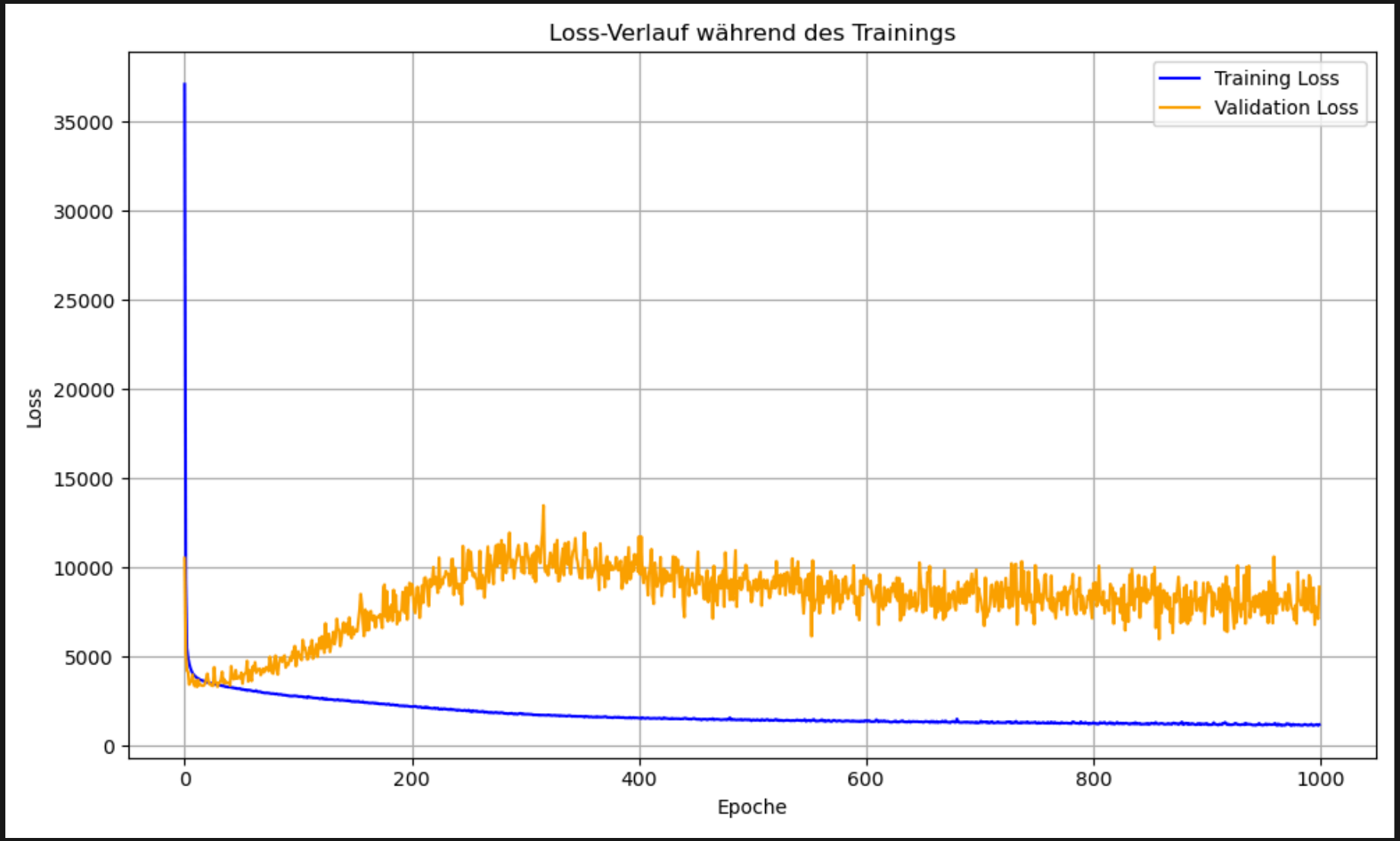
Modell kompilieren

```
model.compile(
    optimizer=optimizer,
    loss='mse',
    metrics=['mae', 'mse']
)
```

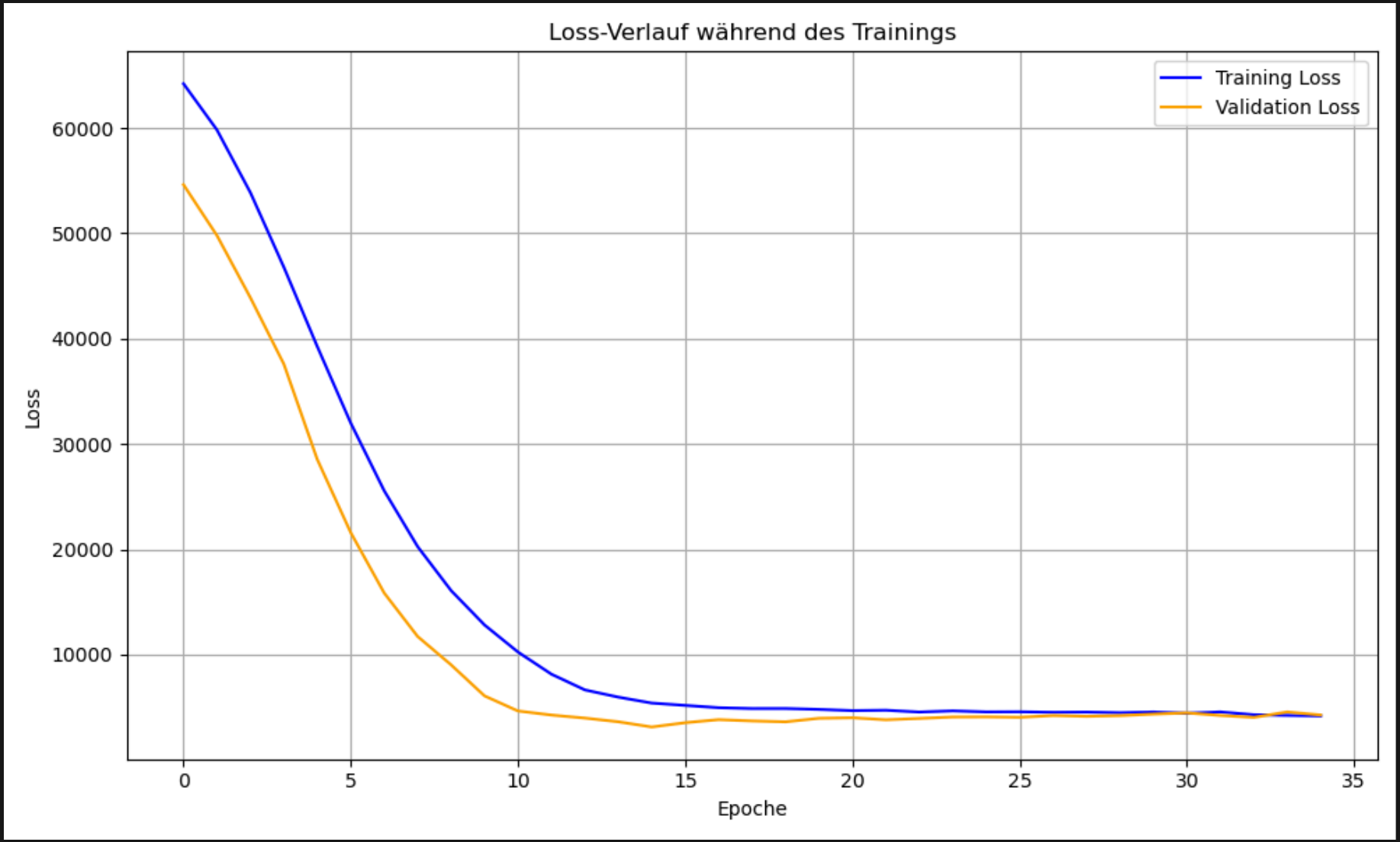
Modell trainieren

```
history = model.fit(
    X_train_scaled, y_train,
    validation_data=(X_val_scaled, y_val),
    epochs=1000,  # Anzahl der Trainingsdurchläufe
    batch_size=32,
    callbacks=[early_stopping, checkpoint],
    verbose=1
)
```

Loss-Funktionen



Loss-Funktionen



MAPEs

Mean Absolute Percentage Error: 20.32%

Warengruppe: 1, MAPE: 22.70%
Warengruppe: 2, MAPE: 16.07%
Warengruppe: 3, MAPE: 18.79%
Warengruppe: 4, MAPE: 21.63%
Warengruppe: 5, MAPE: 18.02%
Warengruppe: 6, MAPE: 52.63%

„Worst Fail“ / „Best Improvement“

- Zeitmangement!
- Problem geeignete Datensätze zu finden
- Theorie verstanden, Schwierigkeit in Code umzusetzen
- Die Vorhersage wie bei Kaggle hinzubekommen
- Repo gelöscht , Codespace weg