Einführung in Data Science & maschinelles Lernen

Gruppe 13
WS 24/25

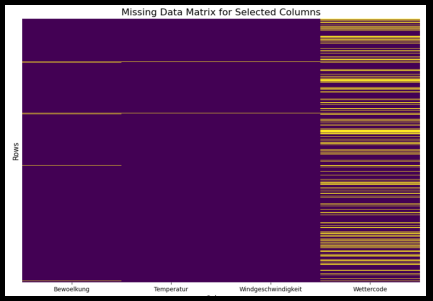
Achraf Aboukinana — Datenzentrierte Optimierung
Christopher Wesemann — Modellzentrierte Optimierung
Agnes Piecyk — PM; Dataset Characteristics

Import der Daten und Vorbereitung

Data Import → Data merging → Handling Missing Values → Data Cleaning → Constructing New Variables

Behandlung fehlender Werte (KNN-Imputation)

Bewoelkung	65
Temperatur	65
Windgeschwindigkeit	65
Wettercode	337




```python
# Gebe die Spalten an, die du imputieren möchtest
spalten_to_impute = ['Bewoelkung', 'Temperatur', 'Windgeschwindigkeit']  # Ersetze mit den tatsächlichen Spaltennamen

# Erstelle eine Kopie des DataFrames, nur mit den gewünschten Spalten für Imputation
df_to_impute = df[spalten_to_impute]

# KNN-Imputation
imputer = KNNImputer(n_neighbors=5)  # Anzahl der Nachbarn für KNN anpassen
df_imputed = pd.DataFrame(imputer.fit_transform(df_to_impute), columns=spalten_to_impute)

# Ersetze die ursprünglichen Werte mit den imputierten Werten
df[spalten_to_impute] = df_imputed

# IDs der Zeilen, die neue Werte erhalten haben
zeilen_mit_neuen_werten = df[df_to_impute.isna().any(axis=1)]['id'].tolist()  # 'id' durch deine ID-Spalte ersetzen

# Speichere den aktualisierten DataFrame in einer neuen Datei
output_file_path = "imputed/imputed_data_test.csv"
df.to_csv(output_file_path, index=False)
```

Selbst erstellte Variablen

Konsum-bezogen

Paycheck effect (wenn 90% der Firmen den Lohn ausgeben)
Inflation sensitivity (high, moderate)
Inflation Kategorisierung (positiv, neutral, negativ)

Tages-bezogen

Feiertag
Ferien
“Special events” (KielLauf, Kieler Triathlon, Fußball)
Wochenende: ja/nein

Wetter-bezogen

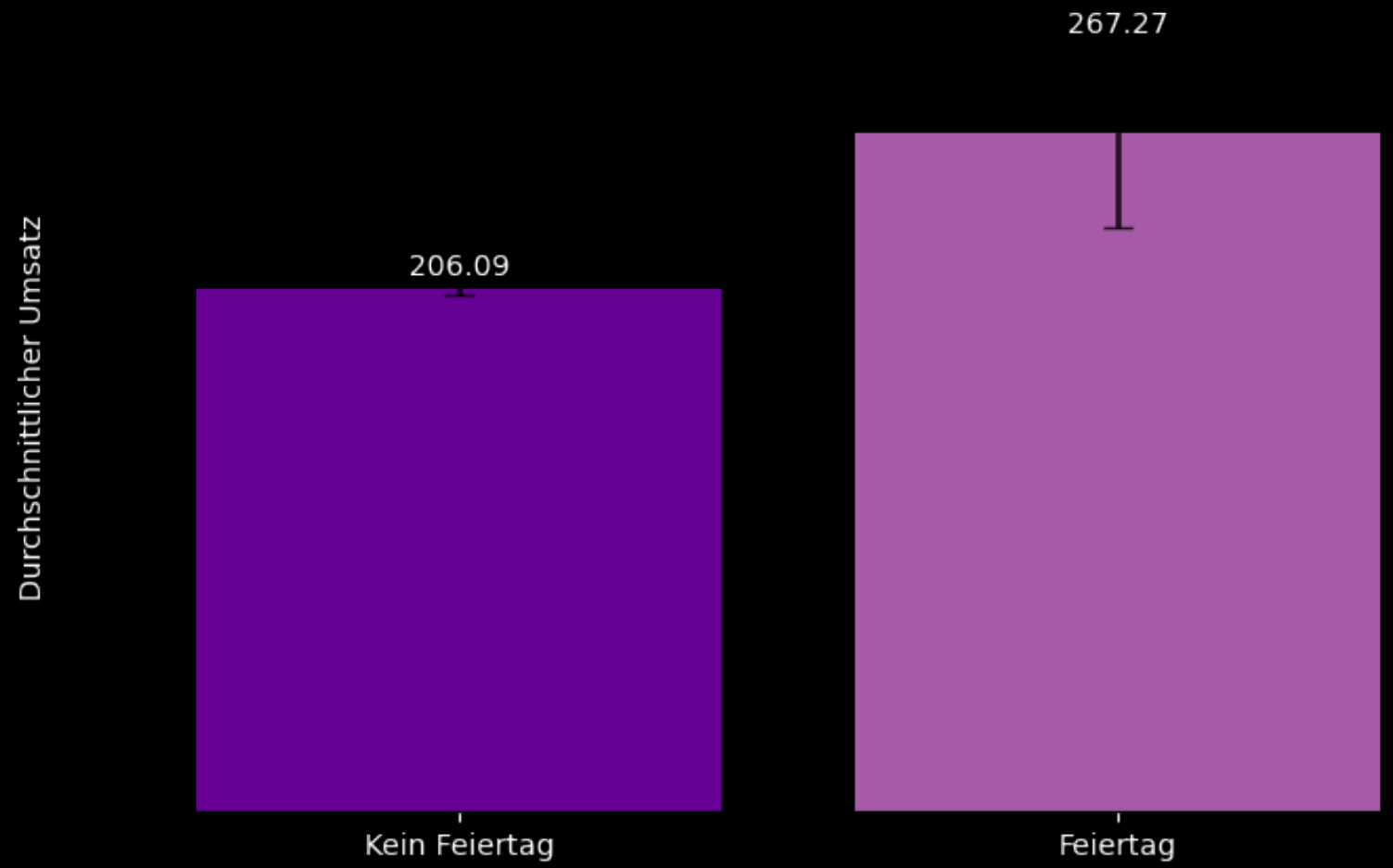
Jahreszeit
Gefühlte Temperatur

Variablen, die wir nicht weiter genutzt haben („worst fail“)

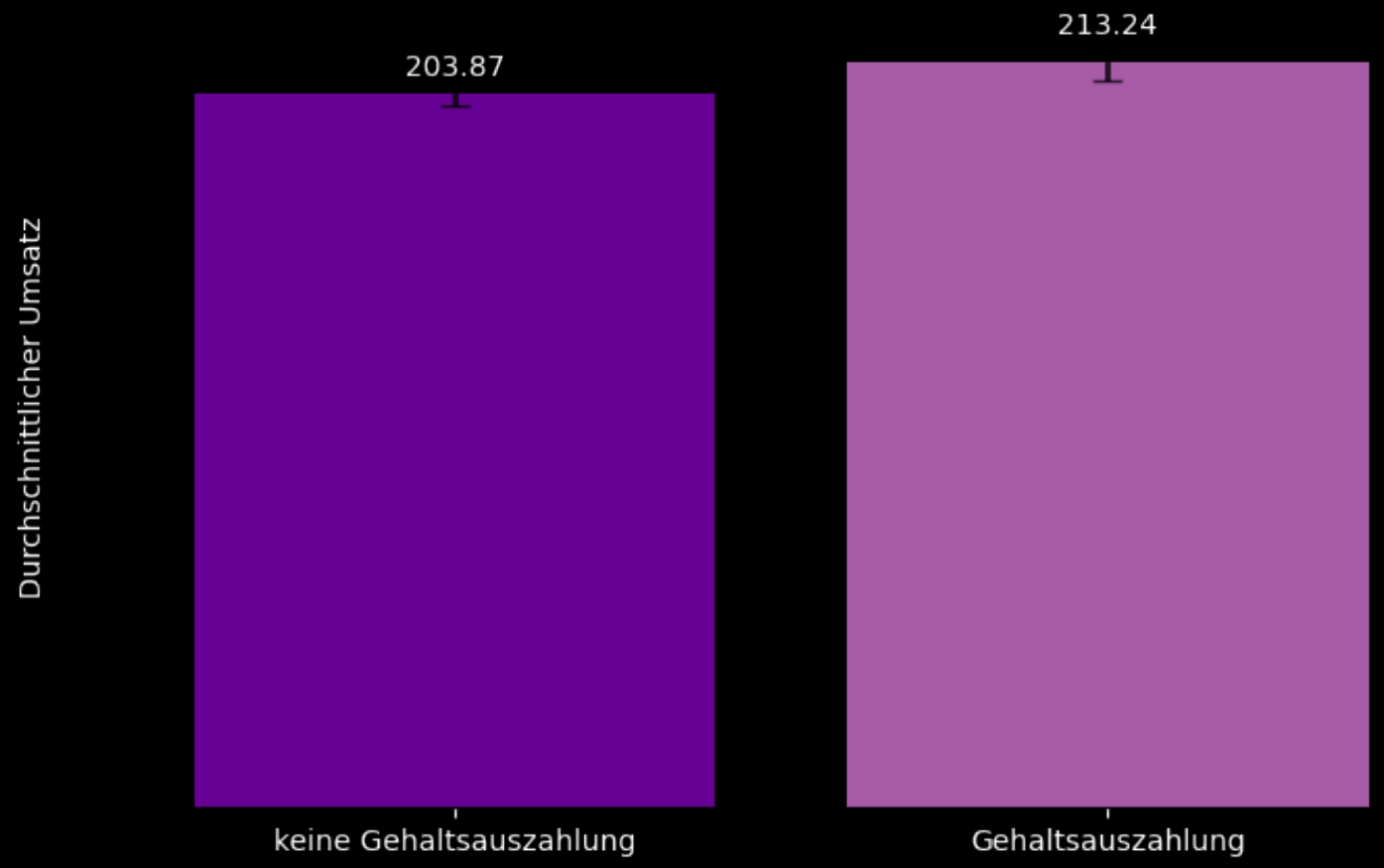
Wetter-bezogen

Wettercode (korreliert mit weiteren Wetter-bezogenen Variablen)

Visuelle Darstellung des Einflusses von Feiertagen



Visuelle Darstellung des Einflusses vom Paycheck Effect



Baseline Model

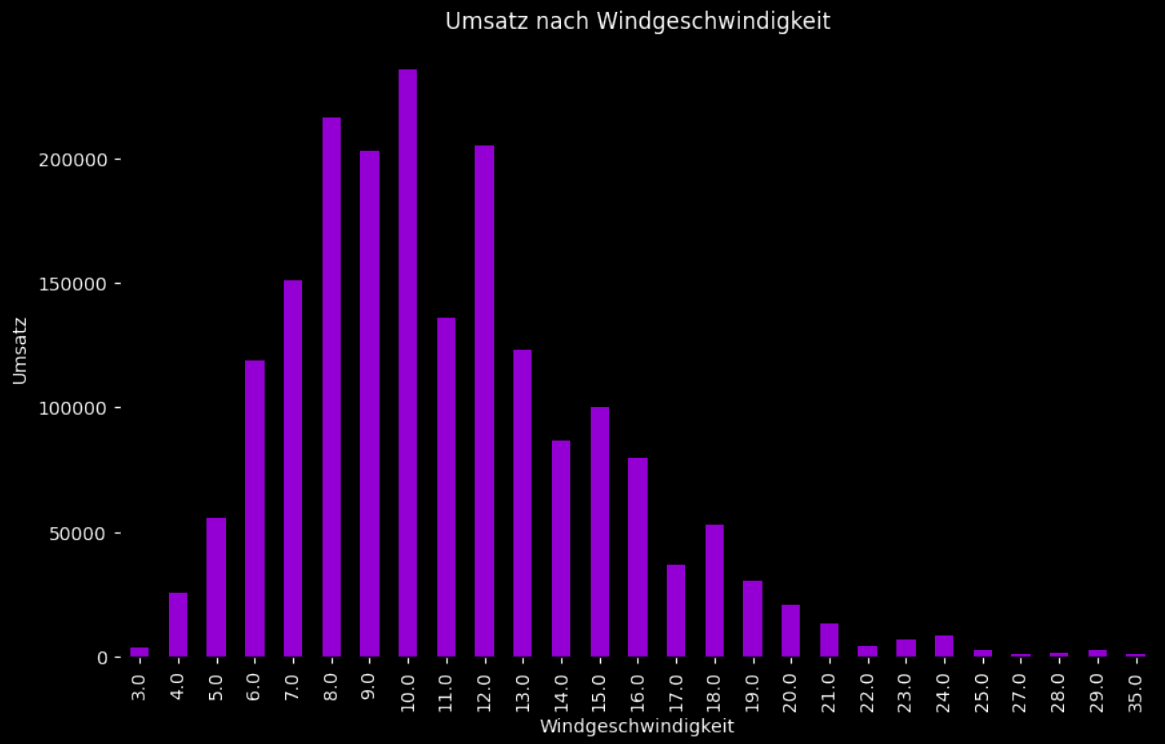
```python
# Features und Ziel definieren
# Beispiel: Ziel ist "Umsatz", Features sind andere numerische Spalten
feature_to_drop= ['id','Umsatz', 'Datum', 'Warengruppe', 'Bewoelkung', 'Temperatur',
       'Windgeschwindigkeit', 'Wettercode',  'Wochentag',
       'Monat', 'Jahr', 'Jahreszeit', 'Gefühl', 'InflationSensitivity',  'Windkategorie',
       'Tag_Kategorie',  'Inflation_Kategorisierung',
       'Montag', 'Dienstag', 'Mittwoch',
       'Donnerstag', 'Freitag', 'Samstag', 'Sonntag']
feature_to_drop_y= ['id', 'Datum', 'Warengruppe', 'Bewoelkung', 'Temperatur',
       'Windgeschwindigkeit', 'Wettercode',  'Wochentag',
       'Monat', 'Jahr', 'Jahreszeit', 'Gefühl', 'InflationSensitivity',  'Windkategorie',
       'Tag_Kategorie',  'Inflation_Kategorisierung',
       'Montag', 'Dienstag', 'Mittwoch',
       'Donnerstag', 'Freitag', 'Samstag', 'Sonntag']


X_train = train_data.drop(columns=feature_to_drop)  # Features: Alle außer 'Umsatz' und 'Datum'

y_train = train_data['Umsatz']  # Ziel: Umsatz

X_test= test_data.drop(columns=feature_to_drop_y)
```

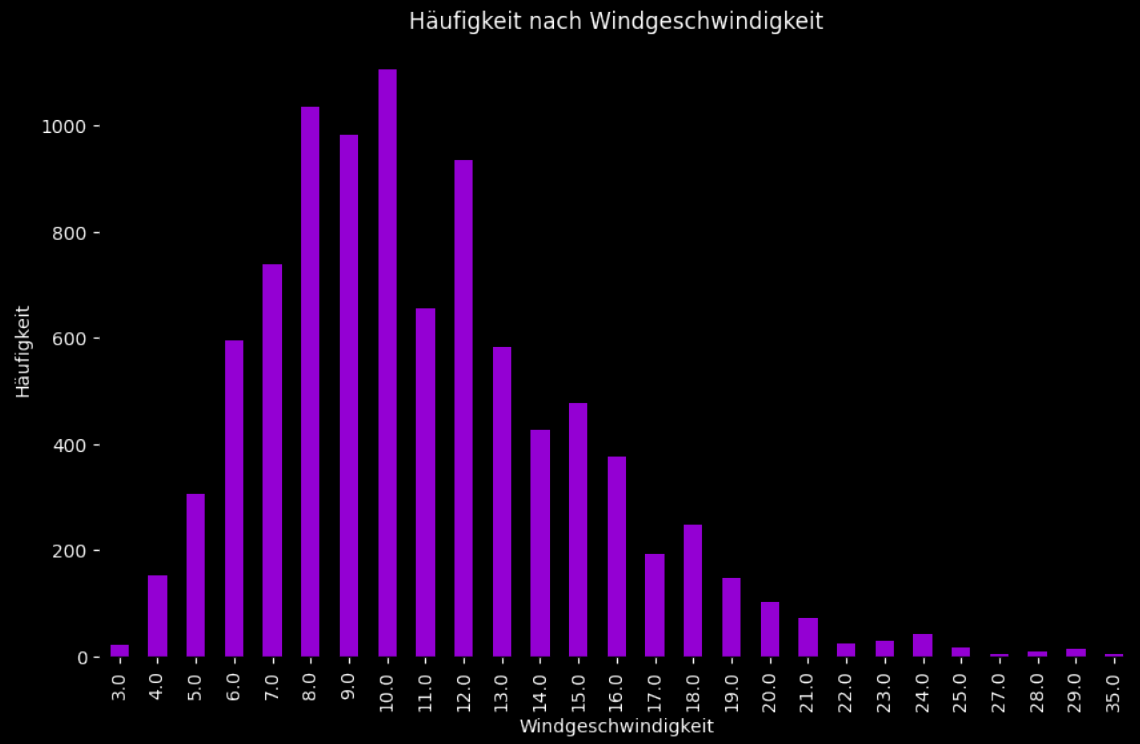
```python
# Train-Test-Split
#X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Lineares Regressionsmodell trainieren
model = LinearRegression()
model.fit(X_train, y_train)

# Vorhersagen
y_pred_train = model.predict(X_train)
y_pred_test = model.predict(X_test)
```

	Feature	Coefficient
13	Wind_Windig	1.251513e+15
11	Wind_Nicht windig	1.251513e+15
12	Wind_Sehr windig	1.251513e+15
10	Sensitivity_Moderate	5.136555e+14
16	Jahreszeit_Sommer	2.309259e+14
15	Jahreszeit_Herbst	2.309259e+14
14	Jahreszeit_Frühling	2.309259e+14
17	Jahreszeit_Winter	2.309259e+14
23	Brötchen	2.004791e+14
22	Brot	2.004791e+14
27	Saisonbrot	2.004791e+14
9	Sensitivity_High	1.133408e+11
4	Feiertag	6.691098e+01
19	Wochenende	5.313209e+01
18	Ferien	3.879072e+01
20	Inflation_Kategorisierung_Neutral	3.031913e+01
5	KielLauf	1.582132e+01
0	KielerWoche	1.113764e+01
6	Kieler_Triathlon	9.193935e+00
21	Inflation_Kategorisierung_Positiv	6.574757e+00
8	PaycheckEffect	4.571864e+00
7	Fußball	-1.725736e+01
3	gefühl_Warm	-1.909840e+13
2	gefühl_Mild	-1.909840e+13
1	gefühl_Kalt	-1.909840e+13
26	Kuchen	-3.130631e+14
24	Croissant	-3.130631e+14
25	Konditorei	-3.130631e+14

```
Trainingsdaten: MSE = 5499.21, R² = 0.74
Anzahl der Features: 28
Anzahl der Koeffizienten: 28
```

Einfluss der Windgeschwindigkeit



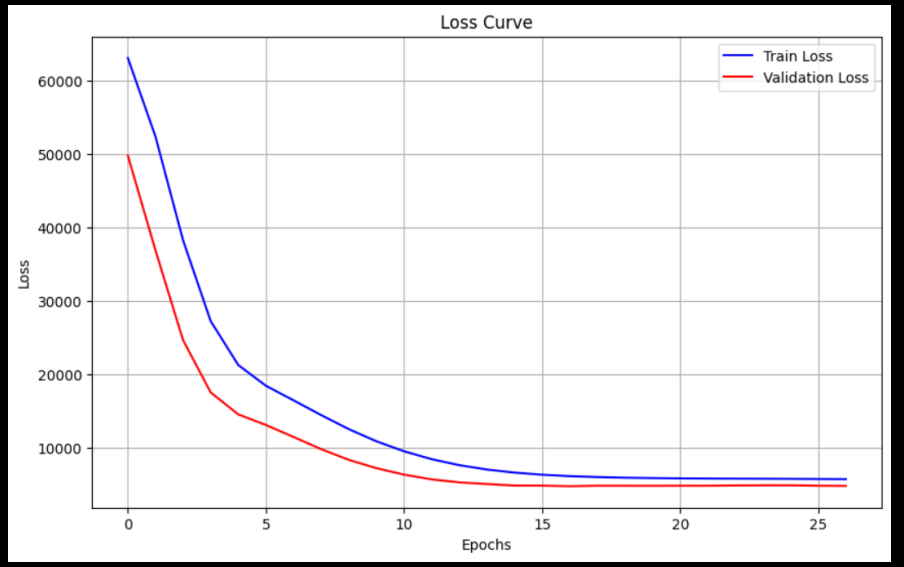


Korrelation (!)

Erstes neuronales Netz

```python
# Eingabedimension
input_dim = X_train.shape[1]

# Modell definieren
model = Sequential()

# Hidden Layer mit 19 Knoten
model.add(Dense(19, input_dim=input_dim, activation='relu'))

# Output Layer
model.add(Dense(1, activation='linear'))

# Modell kompilieren
model.compile(optimizer=Adam(learning_rate=0.001), loss='mse', metrics=['mae'])

# Early Stopping Callback
early_stopping = EarlyStopping(monitor='val_loss', patience=10, restore_best_weights=True)

# Training
history = model.fit(
    X_train, y_train,
    validation_data=(X_val, y_val),
    batch_size=32,
    epochs=100,
    callbacks=[early_stopping]
)
```

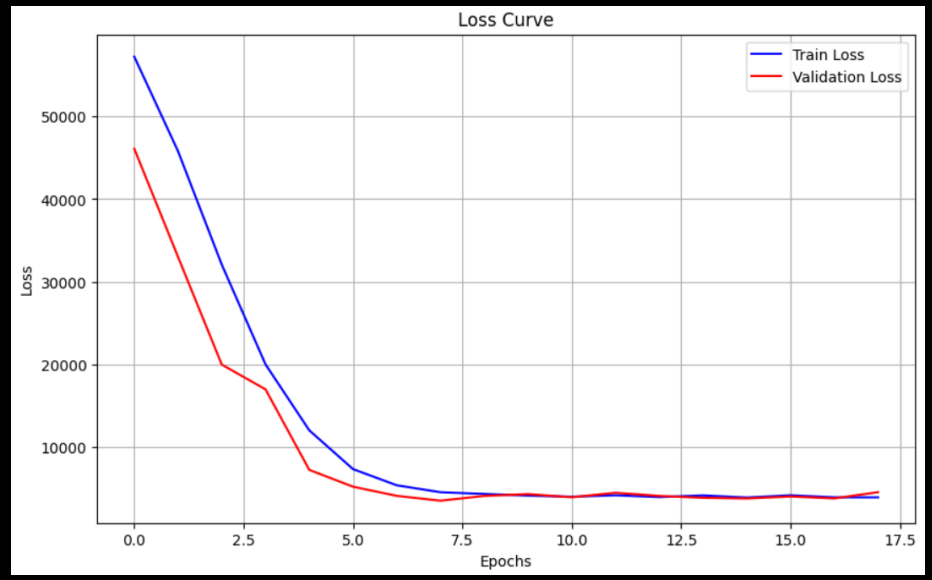
```
MAPE für den Gesamtumsatz pro Tag: 14.8532%
MAPE für den Umsatz pro Warengruppe und Tag:
   Warengruppe        MAPE
0            1   41.377247
1            2   23.812049
2            3   25.497224
3            4   51.767587
4            5   20.166606
5            6  131.229070
Train R²: 0.7231
Validation R²: 0.7286
Train MPAE: 34.2460%
Validation MPAE: 35.4781%
```



Optimierung des neuronalen Netzes

```python
# Eingabedimension
input_dim = X_train.shape[1]

# Modell definieren
model = Sequential()

# Input Layer und Hidden Layer 1
model.add(Dense(128, input_dim=input_dim, activation='relu'))

# Hidden Layer 2
model.add(Dense(256, activation='relu'))
model.add(Dropout(0.3))
model.add(BatchNormalization())

# Hidden Layer 3
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.3))
model.add(BatchNormalization())

# Output Layer
model.add(Dense(1, activation='linear'))

# Modell kompilieren
model.compile(optimizer=Adam(learning_rate=0.001), loss='mse', metrics=['mae'])

# Early Stopping Callback
early_stopping = EarlyStopping(monitor='val_loss', patience=10, restore_best_weights=True)

# Training
history = model.fit(
    X_train, y_train,
    validation_data=(X_val, y_val),
    batch_size=32,
    epochs=100,
```

```
MAPE für den Gesamtumsatz pro Tag: 12.5183%
MAPE für den Umsatz pro Warengruppe und Tag:
   Warengruppe       MAPE
0            1  24.718735
1            2  13.537857
2            3  21.485136
3            4  26.782114
4            5  19.632912
5            6  39.562640
Train R²: 0.8274
Validation R²: 0.7767
Train MPAE: 20.0723%
Validation MPAE: 21.8225%
```



Gegenüberstellung der zwei Modelle

Initiales Model

```
MAPE für den Gesamtumsatz pro Tag: 14.8532%
MAPE für den Umsatz pro Warengruppe und Tag:
   Warengruppe        MAPE
0            1   41.377247
1            2   23.812049
2            3   25.497224
3            4   51.767587
4            5   20.166606
5            6  131.229070
Train R²: 0.7231
Validation R²: 0.7286
Train MPAE: 34.2460%
Validation MPAE: 35.4781%
```

Finales Model

```
MAPE für den Gesamtumsatz pro Tag: 12.5183%
MAPE für den Umsatz pro Warengruppe und Tag:
   Warengruppe        MAPE
0            1   24.718735
1            2   13.537857
2            3   21.485136
3            4   26.782114
4            5   19.632912
5            6   39.562640
Train R²: 0.8274
Validation R²: 0.7767
Train MPAE: 20.0723%
Validation MPAE: 21.8225%
```

Fragen?

Dann immer her damit!