Capstone Projekt Rossmann

XDi - Certified Data Scientist

Christoph Gödecke

Feature Engineering

Functions needed for testing

Test models with test and tain data. Test includes the last 8 weeks from each store

In [26]:
```python
import pandas as pd
import numpy as np
from datetime import datetime

import seaborn as sns
import matplotlib.pyplot as plt
import plotly.express as px
from pandas.api.types import infer_dtype

from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression, Ridge, Lasso
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.svm import SVR
from sklearn.neighbors import KNeighborsRegressor
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.neural_network import MLPRegressor

from xgboost import XGBRegressor
from tensorflow import keras
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout
from scikeras.wrappers import KerasRegressor

from sklearn.metrics import mean_absolute_error as mae, mean_squared_error as mse, r2_score
from math import sqrt

from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import MinMaxScaler
from sklearn.preprocessing import RobustScaler
from sklearn.preprocessing import PolynomialFeatures

pd.set_option('display.max_columns', None)
```

In [27]:
```python
## Test models with test and tain data. Test includes the last 8 weeks from each store

def build_neural_network(X_train):
    model = Sequential()
    model.add(Dense(128, activation='relu', input_dim=X_train.shape[1]))
    model.add(Dropout(0.2))
    model.add(Dense(64, activation='relu'))
    model.add(Dropout(0.2))
    model.add(Dense(1, activation='linear'))
    model.compile(optimizer='adam', loss='mean_squared_error')
    return model





def testModelsTestSplit8W(df, scaler):
        train_data = []
        test_data = []

        # Group by store and split into training and test data
        amount_test_weeks = 8
        for store_id, group in df.groupby('Store'):
                train_data.append(group[: -amount_test_weeks])
                test_data.append(group[-amount_test_weeks:])

        # Combine the list entries to one dataframe
        train_df = pd.concat(train_data)
        test_df = pd.concat(test_data)

        X_train = train_df.drop(columns=['Future_Sales'])
        y_train = train_df['Future_Sales']
        X_test = test_df.drop(columns=['Future_Sales'])
        y_test = test_df['Future_Sales']

        # Scaling of the data
        if scaler:
                X_train = scaler.fit_transform(X_train)
                X_test = scaler.transform(X_test)

        def adj_r2_score(model, X, y):
                n = X.shape[0]
                p = X.shape[1]
                r2 = r2_score(y, model.predict(X))
                return 1 - (1 - r2) * ((n - 1) / (n - p - 1))

        # Defining the models to test
        models = [
                ('LinearRegression', LinearRegression(n_jobs=-1)),
                ('XGBRegressor', XGBRegressor(objective='reg:squarederror', n_estimators=100, max_depth=3, learning_rate=0.1, n_jobs=-1, random_state=42, device="cuda")),
                #('GradientBoostingRegressor', GradientBoostingRegressor(n_estimators=100, learning_rate=0.1, max_depth=3, random_state=42)),
                #('NeuralNetwork', KerasRegressor(build_fn=build_neural_network(X_train), epochs=100, batch_size=10, verbose=0)),
                #('MLPRegressor', MLPRegressor(hidden_layer_sizes=(100,), activation='relu', solver='adam', max_iter=200, shuffle=False, random_state=42))
                #('RidgeRegression', Ridge(random_state=42)),
                #('LassoRegression', Lasso(random_state=42)),
                #('DecisionTreeRegressor', DecisionTreeRegressor(random_state=42)),
                #('RandomForestRegressor', RandomForestRegressor(n_jobs=-1, max_depth=10, random_state=42, n_estimators=100)),
                #('SVR', SVR()),
                #('KNN', KNeighborsRegressor())
        ]

        results = []
        # Train models and calculate metrics
        for name, model in models:
                model.fit(X_train, y_train)
                y_train_pred = model.predict(X_train)
                y_test_pred = model.predict(X_test)

                results.append({
                        'Model': name,
                        'RMSE_Train': sqrt(mse(y_train, y_train_pred)),
                        'MAE_Train': mae(y_train, y_train_pred),
                        'R2_Train': r2_score(y_train, y_train_pred),
```

```
                    'Adj_R2_Train': adj_r2_score(model, X_train, y_train),
                    'RMSE_Test': sqrt(mse(y_test, y_test_pred)),
                    'MAE_Test': mae(y_test, y_test_pred),
                    'R2_Test': r2_score(y_test, y_test_pred),
                    'Adj_R2_Test': adj_r2_score(model, X_test, y_test)
                })
                #print last result
                print(results[-1])

        results_df = pd.DataFrame(results)
        return results_df
```

In [28]:
```
#!pip install scikeras
```

Creates x splits in test and train where the last 8 weeks of each store are included in the respective test split and the splits are distributed evenly using gap

In [29]:
```
#Creates x splits in test and train where the last 8 weeks of each store are included in the respective test split and the splits are distributed
# evenly using gap

def testModelsCV8W(df, scaler):

    n_splits = 5
    window_size = 8
    total_weeks =109
    train_size = window_size / 0.2
    gap = int((total_weeks - window_size - train_size) // (n_splits))

    results = []

    for split in range(n_splits):
        train_data = []
        test_data = []

        for store_id, group in df.groupby('Store'):
            # calculate start and end index for test data
            if split == 0:
                test_start_index = -window_size
                test_df_store = group[test_start_index:] # No end index for the first split
            else:
                test_start_index = -(window_size + gap * split)
                test_end_index = test_start_index + window_size
                test_df_store = group[test_start_index:test_end_index]
                print("test:", test_df_store.shape, "Test Start Index:", test_start_index, "Test End Index:", test_end_index)

            train_start_index = -int(-test_start_index + gap + train_size)
            train_df_store = group[train_start_index:test_start_index]
            print("Train:", train_df_store.shape, "Train Start Index:", train_start_index, "Train End Index:", test_start_index)
            # Check if test set contains data
            if not test_df_store.empty:
                train_data.append(train_df_store)
                test_data.append(test_df_store)
            else:
                print(f"Store {store_id} has not enough data for splitting {split}")

        # Combine the list entries to one dataframe
        train_df_combined = pd.concat(train_data)
        test_df_combined = pd.concat(test_data)

        # Create feature and target data frames
        X_train = train_df_combined.drop(columns=['Future_Sales'])
        y_train = train_df_combined['Future_Sales']
        X_test = test_df_combined.drop(columns=['Future_Sales'])
        y_test = test_df_combined['Future_Sales']

        # Scaling of the data
        if scaler:
            X_train = scaler.fit_transform(X_train)
            X_test = scaler.transform(X_test)

        def adj_r2_score(model, X, y):
            n = X.shape[0]
            p = X.shape[1]
            r2 = r2_score(y, model.predict(X))
            return 1 - (1 - r2) * ((n - 1) / (n - p - 1))

        # Defining the models to test
        models = [
            ('LinearRegression', LinearRegression(n_jobs=-1)),
            #('RidgeRegression', Ridge(random_state=42)),
            #('LassoRegression', Lasso(random_state=42)),
            #('DecisionTreeRegressor', DecisionTreeRegressor(random_state=42)),
            #('RandomForestRegressor', RandomForestRegressor(n_jobs=-1, max_depth=10, random_state=42, n_estimators=100)),
            #('SVR', SVR()),
            #('KNN', KNeighborsRegressor())
        ]

        # Train models and calculate metrics
        for name, model in models:
            model.fit(X_train, y_train)
            y_train_pred = model.predict(X_train)
            y_test_pred = model.predict(X_test)

            results.append({
                'Model': name,
                'RMSE_Train': sqrt(mse(y_train, y_train_pred)),
                'MAE_Train': mae(y_train, y_train_pred),
                'R2_Train': r2_score(y_train, y_train_pred),
                'Adj_R2_Train': adj_r2_score(model, X_train, y_train),
                'RMSE_Test': sqrt(mse(y_test, y_test_pred)),
                'MAE_Test': mae(y_test, y_test_pred),
                'R2_Test': r2_score(y_test, y_test_pred),
                'Adj_R2_Test': adj_r2_score(model, X_test, y_test)
            })
            #print last result
            print(results[-1])

    results_df = pd.DataFrame(results)

    # calculate mean of all splits
    model_list = results_df['Model'].unique()
    # create resulte_mean_df
    resulte_mean_df = pd.DataFrame(columns=results_df.columns)
    # iterate over model_list
    for model in model_list:
        # get mean of each model
        mean = results_df[results_df['Model'] == model].mean(numeric_only=True)
        mean['Model'] = model
        # append mean to resulte_mean_df
        resulte_mean_df = pd.concat([resulte_mean_df, pd.DataFrame([mean], columns=results_df.columns)], ignore_index=True)

    return results_df, resulte_mean_df
```

Feature Engineering

In [30]:
```
df = pd.read_csv('weekly_sales_with_store_info.csv')
df['Date'] = pd.to_datetime(df['Date'])
```

In [31]:
```
print(df.info())
df.sample(5)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150525 entries, 0 to 150524
Data columns (total 31 columns):
 #   Column                     Non-Null Count   Dtype
---  ------                     --------------   -----
 0   Store                      150525 non-null  int64
 1   Date                       150525 non-null  datetime64[ns]
 2   CW                         150525 non-null  int64
 3   Month                      150525 non-null  int64
 4   Year                       150525 non-null  int64
 5   DayOfWeek                  150525 non-null  int64
 6   Sales                      150525 non-null  int64
 7   SalesPerCustomer           145809 non-null  float64
 8   SalesPerOpenDay            145815 non-null  float64
 9   Customers                  150525 non-null  int64
 10  CustomersPerOpenDay        145815 non-null  float64
 11  Open                       150525 non-null  int64
 12  Promo                      150525 non-null  int64
 13  IsPromo                    150525 non-null  int64
 14  StateHoliday               150525 non-null  object
 15  IsStateHoliday             150525 non-null  int64
 16  SchoolHoliday              150525 non-null  int64
 17  IsSchoolHoliday            150525 non-null  int64
 18  NumStateHoliday            150525 non-null  int64
 19  StoreType                  150525 non-null  object
 20  Assortment                 150525 non-null  object
 21  CompetitionDistance        150120 non-null  float64
 22  CompetitionOpenSinceMonth  102735 non-null  float64
 23  CompetitionOpenSinceYear   102735 non-null  float64
 24  IsCompetition              150525 non-null  int64
 25  Promo2                     150525 non-null  int64
 26  Promo2SinceWeek            77085 non-null   float64
 27  Promo2SinceYear            77085 non-null   float64
 28  PromoInterval              77085 non-null   object
 29  Promo2Member               150525 non-null  int64
 30  Promo2Active               150525 non-null  int64
dtypes: datetime64[ns](1), float64(8), int64(18), object(4)
memory usage: 35.6+ MB
None
```

Out[31]:

	Store	Date	CW	Month	Year	DayOfWeek	Sales	SalesPerCustomer	SalesPerOpenDay	Customers	CustomersPerOpenDay	Open	Promo	IsPromo	StateHoliday	IsStateHoliday	SchoolHoliday	IsSchoolHoliday
137011	1015	2015-05-03	18	5	2015	6	38446	13.924665	7689.200000	2761	552.200000	5	5	1	a	1	0	0
7451	56	2013-07-07	27	7	2013	6	48251	15.727184	8041.833333	3068	511.333333	6	5	1	0	0	2	1
81710	606	2013-09-08	36	9	2013	6	28173	8.545041	4695.500000	3297	549.500000	6	0	0	0	0	0	0
76851	570	2013-09-15	37	9	2013	6	27903	6.715523	4650.500000	4155	692.500000	6	5	1	0	0	0	0
58904	437	2013-11-10	45	11	2013	6	46248	10.300223	7708.000000	4490	748.333333	6	5	1	0	0	0	0

Handle Missing Values

In [32]:
```
df.isna().sum()
```

Out[32]:
```
Store                            0
Date                             0
CW                               0
Month                            0
Year                             0
DayOfWeek                        0
Sales                            0
SalesPerCustomer              4716
SalesPerOpenDay               4710
Customers                        0
CustomersPerOpenDay           4710
Open                             0
Promo                            0
IsPromo                          0
StateHoliday                     0
IsStateHoliday                   0
SchoolHoliday                    0
IsSchoolHoliday                  0
NumStateHoliday                  0
StoreType                        0
Assortment                       0
CompetitionDistance            405
CompetitionOpenSinceMonth    47790
CompetitionOpenSinceYear     47790
IsCompetition                    0
Promo2                           0
Promo2SinceWeek              73440
Promo2SinceYear              73440
PromoInterval                73440
Promo2Member                     0
Promo2Active                     0
dtype: int64
```

SalesPerCustomer, SalesPerOpenday, CustomersPerOpenday

In [33]:
```
# As the store were closed, we can fill the nans with 0

# fill nans with 0 for listed columns
columns_to_fill = ['SalesPerCustomer', 'SalesPerOpenDay', 'CustomersPerOpenDay']
df_nans_handeled = df.fillna({col: 0 for col in columns_to_fill})
df_nans_handeled
```

Out[33]:

	Store	Date	CW	Month	Year	DayOfWeek	Sales	SalesPerCustomer	SalesPerOpenDay	Customers	CustomersPerOpenDay	Open	Promo	IsPromo	StateHoliday	IsStateHoliday	SchoolHoliday	IsSchoolHoliday
0	1	2013-01-06	1	1	2013	6	19340	7.736000	4835.000000	2500	625.000000	4	0	0	a	1	6	1
1	1	2013-01-13	2	1	2013	6	32952	8.410413	5492.000000	3918	653.000000	6	5	1	0	0	5	1
2	1	2013-01-20	3	1	2013	6	25978	7.602575	4329.666667	3417	569.500000	6	0	0	0	0	0	0
3	1	2013-01-27	4	1	2013	6	33071	8.563180	5511.833333	3862	643.666667	6	5	1	0	0	0	0
4	1	2013-02-03	5	2	2013	6	28693	8.057568	4782.166667	3561	593.500000	6	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
150520	1115	2015-07-05	27	7	2015	6	48130	16.140174	8021.666667	2982	497.000000	6	5	1	0	0	0	0
150521	1115	2015-07-12	28	7	2015	6	36233	14.315685	6038.833333	2531	421.833333	6	0	0	0	0	0	0
150522	1115	2015-07-19	29	7	2015	6	45927	15.023553	7654.500000	3057	509.500000	6	5	1	0	0	0	0
150523	1115	2015-07-26	30	7	2015	6	35362	14.122204	5893.666667	2504	417.333333	6	0	0	0	0	0	0
150524	1115	2015-08-02	31	8	2015	6	43551	16.616177	8710.200000	2621	524.200000	5	5	1	0	0	5	1

150525 rows × 31 columns

In [34]:

```
df_nans_handeled.isna().sum()
```

Out[34]:

```
Store                          0
Date                           0
CW                             0
Month                          0
Year                           0
DayOfWeek                      0
Sales                          0
SalesPerCustomer               0
SalesPerOpenDay                0
Customers                      0
CustomersPerOpenDay            0
Open                           0
Promo                          0
IsPromo                        0
StateHoliday                   0
IsStateHoliday                 0
SchoolHoliday                  0
IsSchoolHoliday                0
NumStateHoliday                0
StoreType                      0
Assortment                     0
CompetitionDistance          405
CompetitionOpenSinceMonth    47790
CompetitionOpenSinceYear     47790
IsCompetition                  0
Promo2                         0
Promo2SinceWeek            73440
Promo2SinceYear            73440
PromoInterval              73440
Promo2Member                   0
Promo2Active                   0
dtype: int64
```

CompetitionDistance

In [35]:

```
# Stores with no CompetitionDistance information
print("Stores with no CompetitionDistance information:", df_nans_handeled[(df_nans_handeled['CompetitionDistance'].isna())]['Store'].unique())

print("StoreType of store 291", df_nans_handeled[(df_nans_handeled['Store'] == 291)]['StoreType'].unique())
print("StoreType of store 622", df_nans_handeled[(df_nans_handeled['Store'] == 622)]['StoreType'].unique())
print("StoreType of store 879", df_nans_handeled[(df_nans_handeled['Store'] == 879)]['StoreType'].unique())
```

```
Stores with no CompetitionDistance information: [291 622 879]
StoreType of store 291 ['d']
StoreType of store 622 ['a']
StoreType of store 879 ['d']
```

In [36]:

```
# As store 291, 622 and 879 have no CompetitionDistance information, we can fill them with the median value of the column

# median competition distance for store type a
median_competition_distance_a = df_nans_handeled[(df_nans_handeled['StoreType'] == 'a')]['CompetitionDistance'].median()
# median competition distance for store type d
median_competition_distance_d = df_nans_handeled[(df_nans_handeled['StoreType'] == 'd')]['CompetitionDistance'].median()

# fill nans for storetype a with median_competition_distance_a
df_nans_handeled.loc[(df_nans_handeled['Store'] == 291), 'CompetitionDistance'] = median_competition_distance_a
# fill nans for storetype d with median_competition_distance_d
df_nans_handeled.loc[(df_nans_handeled['Store'] == 622), 'CompetitionDistance'] = median_competition_distance_d
# fill nans for storetype d with median_competition_distance_d
df_nans_handeled.loc[(df_nans_handeled['Store'] == 879), 'CompetitionDistance'] = median_competition_distance_d
```

In [37]:

```
df_nans_handeled.isna().sum()
```

Out[37]:
```
Store                          0
Date                           0
CW                             0
Month                          0
Year                           0
DayOfWeek                      0
Sales                          0
SalesPerCustomer               0
SalesPerOpenDay                0
Customers                      0
CustomersPerOpenDay            0
Open                           0
Promo                          0
IsPromo                        0
StateHoliday                   0
IsStateHoliday                 0
SchoolHoliday                  0
IsSchoolHoliday                0
NumStateHoliday                0
StoreType                      0
Assortment                     0
CompetitionDistance            0
CompetitionOpenSinceMonth  47790
CompetitionOpenSinceYear   47790
IsCompetition                  0
Promo2                         0
Promo2SinceWeek            73440
Promo2SinceYear            73440
PromoInterval              73440
Promo2Member                   0
Promo2Active                   0
dtype: int64
```

CompetitionOpenSinceMonth, CompetitionOpenSinceYear

In [38]:
```
# CompetitionOpenSinceMonth and CompetitionOpenSinceYear can be deleted as they are reflected in IsCompetition
df_nans_handeled = df_nans_handeled.drop(columns=['CompetitionOpenSinceMonth', 'CompetitionOpenSinceYear'])
```

In [39]:
```
df_nans_handeled.isna().sum()
```

Out[39]:
```
Store                      0
Date                       0
CW                         0
Month                      0
Year                       0
DayOfWeek                  0
Sales                      0
SalesPerCustomer           0
SalesPerOpenDay            0
Customers                  0
CustomersPerOpenDay        0
Open                       0
Promo                      0
IsPromo                    0
StateHoliday               0
IsStateHoliday             0
SchoolHoliday              0
IsSchoolHoliday            0
NumStateHoliday            0
StoreType                  0
Assortment                 0
CompetitionDistance        0
IsCompetition              0
Promo2                     0
Promo2SinceWeek        73440
Promo2SinceYear        73440
PromoInterval          73440
Promo2Member               0
Promo2Active               0
dtype: int64
```

Promo2SinceWeek, Promo2SinceYear

In [40]:
```
# Promo2SinceWeek and Promo2SinceYear can be deleted as they are reflected in Promo2Member
df_nans_handeled = df_nans_handeled.drop(columns=['Promo2SinceWeek', 'Promo2SinceYear'])
```

In [41]:
```
df_nans_handeled.isna().sum()
```

Out[41]:
```
Store                      0
Date                       0
CW                         0
Month                      0
Year                       0
DayOfWeek                  0
Sales                      0
SalesPerCustomer           0
SalesPerOpenDay            0
Customers                  0
CustomersPerOpenDay        0
Open                       0
Promo                      0
IsPromo                    0
StateHoliday               0
IsStateHoliday             0
SchoolHoliday              0
IsSchoolHoliday            0
NumStateHoliday            0
StoreType                  0
Assortment                 0
CompetitionDistance        0
IsCompetition              0
Promo2                     0
PromoInterval          73440
Promo2Member               0
Promo2Active               0
dtype: int64
```

PromoInterval

In [42]:
```
df_nans_handeled[(df_nans_handeled['Promo2'] == 1) & (df_nans_handeled['PromoInterval'].isna())]
```

Out[42]:

Store	Date	CW	Month	Year	DayOfWeek	Sales	SalesPerCustomer	SalesPerOpenDay	Customers	CustomersPerOpenDay	Open	Promo	IsPromo	StateHoliday	IsStateHoliday	SchoolHoliday	IsSchoolHoliday	NumState

In [43]:
```
# As if the store is not participating in Promo2, PromoInterval is 0, we can fill the nans with 0
df_nans_handeled['PromoInterval'] = df_nans_handeled['PromoInterval'].fillna(0)
```

In [44]:
```
df_nans_handeled.isna().sum()
```

Out[44]:
```
Store                  0
Date                   0
CW                     0
Month                  0
Year                   0
DayOfWeek              0
Sales                  0
SalesPerCustomer       0
SalesPerOpenDay        0
Customers              0
CustomersPerOpenDay    0
Open                   0
Promo                  0
IsPromo                0
StateHoliday           0
IsStateHoliday         0
SchoolHoliday          0
IsSchoolHoliday        0
NumStateHoliday        0
StoreType              0
Assortment             0
CompetitionDistance    0
IsCompetition          0
Promo2                 0
PromoInterval          0
Promo2Member           0
Promo2Active           0
dtype: int64
```

Remove not needed Features

In [45]:
```python
# Date is an object and is reflected by CW, Month and Year
df_nans_handeled = df_nans_handeled.drop(columns=['Date'])

# DayOfWeek is not relevant in weekly data
df_nans_handeled = df_nans_handeled.drop(columns=['DayOfWeek'])

df_nans_handeled
```

Out[45]:

	Store	CW	Month	Year	Sales	SalesPerCustomer	SalesPerOpenDay	Customers	CustomersPerOpenDay	Open	Promo	IsPromo	StateHoliday	IsStateHoliday	SchoolHoliday	IsSchoolHoliday	NumStateHoliday	St
0	1	1	1	2013	19340	7.736000	4835.000000	2500	625.000000	4	0	0	a	1	6	1	1	
1	1	2	1	2013	32952	8.410413	5492.000000	3918	653.000000	6	5	1	0	0	5	1	0	
2	1	3	1	2013	25978	7.602575	4329.666667	3417	569.500000	6	0	0	0	0	0	0	0	
3	1	4	1	2013	33071	8.563180	5511.833333	3862	643.666667	6	5	1	0	0	0	0	0	
4	1	5	2	2013	28693	8.057568	4782.166667	3561	593.500000	6	0	0	0	0	0	0	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
150520	1115	27	7	2015	48130	16.140174	8021.666667	2982	497.000000	6	5	1	0	0	0	0	0	
150521	1115	28	7	2015	36233	14.315685	6038.833333	2531	421.833333	6	0	0	0	0	0	0	0	
150522	1115	29	7	2015	45927	15.023553	7654.500000	3057	509.500000	6	5	1	0	0	0	0	0	
150523	1115	30	7	2015	35362	14.122204	5893.666667	2504	417.333333	6	0	0	0	0	0	0	0	
150524	1115	31	8	2015	43551	16.616177	8710.200000	2621	524.200000	5	5	1	0	0	5	1	0	

150525 rows × 25 columns

Categorical Feature Encoding

In [46]:
```python
df_nans_handeled.select_dtypes(include='object').columns
```

Out[46]:
```
Index(['StateHoliday', 'StoreType', 'Assortment', 'PromoInterval'], dtype='object')
```

In [47]:
```python
# check if a column contains mixed data typesrom pandas.api.types import infer_dtype

for col in ['StateHoliday', 'StoreType', 'Assortment', 'PromoInterval']:
    dtype = infer_dtype(df_nans_handeled[col])
    print(f"Data type of {col}: {dtype}")
```

```
Data type of StateHoliday: string
Data type of StoreType: string
Data type of Assortment: string
Data type of PromoInterval: mixed-integer
```

In [48]:
```python
# Convert mixed columns
cols_to_convert = ['PromoInterval']
df_nans_handeled[cols_to_convert] = df_nans_handeled[cols_to_convert].astype(str)
```

In [49]:
```python
from sklearn.preprocessing import OneHotEncoder
# handle_unknown='ignore': to avoid error if the training data contains classes/categories that are not represented in the training data
# sparse=False: ensures that the encoded columns are returned as a NumPy array (instead of a sparse matrix).
OneHotEnc = OneHotEncoder(handle_unknown='ignore', sparse_output=False)
# it is important to pass only the categorical columns, not the whole dataframe
encoded_array = OneHotEnc.fit_transform(df_nans_handeled[['StateHoliday', 'StoreType', 'Assortment', 'PromoInterval']])
tmp_cat = pd.DataFrame(encoded_array, columns=OneHotEnc.get_feature_names_out(), index=df_nans_handeled.index)
df_nans_handeled_cat = pd.concat([df_nans_handeled.select_dtypes(include=['number']), tmp_cat], axis=1)
df_nans_handeled_cat
```

Out[49]:

	Store	CW	Month	Year	Sales	SalesPerCustomer	SalesPerOpenDay	Customers	CustomersPerOpenDay	Open	Promo	IsPromo	IsStateHoliday	SchoolHoliday	IsSchoolHoliday	NumStateHoliday	CompetitionDista
0	1	1	1	2013	19340	7.736000	4835.000000	2500	625.000000	4	0	0	1	6	1	1	12
1	1	2	1	2013	32952	8.410413	5492.000000	3918	653.000000	6	5	1	0	5	1	0	12
2	1	3	1	2013	25978	7.602575	4329.666667	3417	569.500000	6	0	0	0	0	0	0	12
3	1	4	1	2013	33071	8.563180	5511.833333	3862	643.666667	6	5	1	0	0	0	0	12
4	1	5	2	2013	28693	8.057568	4782.166667	3561	593.500000	6	0	0	0	0	0	0	12
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
150520	1115	27	7	2015	48130	16.140174	8021.666667	2982	497.000000	6	5	1	0	0	0	0	53
150521	1115	28	7	2015	36233	14.315685	6038.833333	2531	421.833333	6	0	0	0	0	0	0	53
150522	1115	29	7	2015	45927	15.023553	7654.500000	3057	509.500000	6	5	1	0	0	0	0	53
150523	1115	30	7	2015	35362	14.122204	5893.666667	2504	417.333333	6	0	0	0	0	0	0	53
150524	1115	31	8	2015	43551	16.616177	8710.200000	2621	524.200000	5	5	1	0	5	1	0	53

150525 rows × 36 columns

In [50]:
```python
# Creating Lag-Features for these columns
lag_columns = ['Sales', 'SalesPerCustomer', 'SalesPerOpenDay', 'Customers', 'CustomersPerOpenDay']
n_lags = 8  # Anzahl der zu erstellenden Lag-Features
n_periods_for_ma = 4  # amount of periods for moving average
n_periods_for_ma2 = 6
n_periods_for_ma3 = 8
```

```
lag_features = []
for col in lag_columns:
    store_groups = df_nans_handeled_cat.groupby('Store')[col]
    for lag in range(1, n_lags + 1):
        lag_col_name = f'{col}_Lag_{lag}'
        # create the lag feature
        lag_feature = store_groups.shift(lag).rename(lag_col_name)
        lag_features.append(lag_feature)

        # create the moving average feature
        ma_col_name = f'{lag_col_name}_MA_{n_periods_for_ma}'
        ma_feature = lag_feature.rolling(window=n_periods_for_ma).mean().rename(ma_col_name)
        lag_features.append(ma_feature)

        ma2_col_name = f'{lag_col_name}_MA_{n_periods_for_ma2}'
        ma2_feature = lag_feature.rolling(window=n_periods_for_ma2).mean().rename(ma2_col_name)
        lag_features.append(ma2_feature)

        ma3_col_name = f'{lag_col_name}_MA_{n_periods_for_ma3}'
        ma3_feature = lag_feature.rolling(window=n_periods_for_ma3).mean().rename(ma3_col_name)
        lag_features.append(ma3_feature)

# Concatenate the lag features
features_df = pd.concat(lag_features, axis=1)
df_nans_handeled_cat = pd.concat([df_nans_handeled_cat, features_df], axis=1)

# Add the future sales
future_sales = df_nans_handeled_cat.groupby('Store')['Sales'].shift(-8).rename('Future_Sales')
df_nans_handeled_cat = pd.concat([df_nans_handeled_cat, future_sales], axis=1)

# Remove rows with NaN values that were created by the shifting and remove the original columns
df_nans_handeled_cat = df_nans_handeled_cat.dropna().drop(columns=lag_columns)
df_nans_handeled_cat
```

Out[50]:

	Store	CW	Month	Year	Open	Promo	IsPromo	IsStateHoliday	SchoolHoliday	IsSchoolHoliday	NumStateHoliday	CompetitionDistance	IsCompetition	Promo2	Promo2Member	Promo2Active	StateHoliday_0
15	1	16	4	2013	6	0	0	0	0	0	0	1270.0	1	0	0	0	1.0
16	1	17	4	2013	6	5	1	0	0	0	0	1270.0	1	0	0	0	1.0
17	1	18	5	2013	5	5	1	1	0	0	1	1270.0	1	0	0	0	0.0
18	1	19	5	2013	5	0	0	1	0	0	1	1270.0	1	0	0	0	0.0
19	1	20	5	2013	6	5	1	0	0	0	0	1270.0	1	0	0	0	1.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
150512	1115	19	5	2015	6	5	1	0	0	0	0	5350.0	0	1	1	0	1.0
150513	1115	20	5	2015	5	0	0	1	0	0	1	5350.0	0	1	1	0	0.0
150514	1115	21	5	2015	6	5	1	0	0	0	0	5350.0	0	1	1	0	1.0
150515	1115	22	5	2015	5	0	0	1	0	0	1	5350.0	0	1	1	0	0.0
150516	1115	23	6	2015	5	5	1	1	0	0	1	5350.0	0	1	1	1	0.0

124880 rows × 192 columns

New additional Features

Ploynominal Features

In [51]:
```
# Before
testModelsTestSplit8W(df_nans_handeled_cat, None)
```

```
{'Model': 'LinearRegression', 'RMSE_Train': 9348.624262728845, 'MAE_Train': 6448.18666177155, 'R2_Train': 0.7367209231767318, 'Adj_R2_Train': 0.7362865518161379, 'RMSE_Test': 6029.131809472689, 'MAE_Test': 4495.405879127806, 'R2_Test': 0.8600745236580398, 'Adj_R2_Test': 0.8570124514786959}
c:\Users\Chris\anaconda3\lib\site-packages\xgboost\core.py:160: UserWarning: [22:04:21] WARNING: C:\buildkite-agent\builds\buildkite-windows-cpu-autoscaling-group-i-0b3782d1791676daf-1\xgboost\xgboost-ci-windows\src\context.cc:44: No visible GPU is found, setting device to CPU.
  warnings.warn(smsg, UserWarning)
{'Model': 'XGBRegressor', 'RMSE_Train': 7410.980873242441, 'MAE_Train': 5030.000189809052, 'R2_Train': 0.8345478857952568, 'Adj_R2_Train': 0.8342749143885373, 'RMSE_Test': 6058.148984876124, 'MAE_Test': 4515.474892052621, 'R2_Test': 0.858724407972136, 'Adj_R2_Test': 0.8556327904105729}
```

Out[51]:

	Model	RMSE_Train	MAE_Train	R2_Train	Adj_R2_Train	RMSE_Test	MAE_Test	R2_Test	Adj_R2_Test
0	LinearRegression	9348.624263	6448.186662	0.736721	0.736287	6029.131809	4495.405879	0.860075	0.857012
1	XGBRegressor	7410.980873	5030.000190	0.834548	0.834275	6058.148985	4515.474892	0.858724	0.855633

In [54]:
```
pf = PolynomialFeatures(degree=2)
features = ['Open', 'Promo', 'IsPromo', 'IsStateHoliday', 'SchoolHoliday', 'IsSchoolHoliday', 'CompetitionDistance', 'IsCompetition', 'Promo2Member', 'Assortment_a', 'Assortment_b', 'Assortment_c']

feat_array = pf.fit_transform(df_nans_handeled_cat[features])
# generate names for the new features
feature_names = pf.get_feature_names_out(input_features=features)
poly_features_df = pd.DataFrame(feat_array, columns=feature_names, index=df_nans_handeled_cat.index)
# remove the '1' column
poly_features_df = poly_features_df.drop('1', axis=1)
# Concatenate the polynomial features with the original DataFrame
df_nans_handeled_cat_poly = pd.concat([df_nans_handeled_cat.drop(features, axis=1), poly_features_df], axis=1)
testModelsTestSplit8W(df_nans_handeled_cat_poly, None)
```

```
{'Model': 'LinearRegression', 'RMSE_Train': 9196.055248192532, 'MAE_Train': 6324.87532423575, 'R2_Train': 0.7452442003080441, 'Adj_R2_Train': 0.7446518473811088, 'RMSE_Test': 6753.577695278919, 'MAE_Test': 5219.836166297047, 'R2_Test': 0.8244280883192745, 'Adj_R2_Test': 0.8189681063259664}
c:\Users\Chris\anaconda3\lib\site-packages\xgboost\core.py:160: UserWarning: [22:12:31] WARNING: C:\buildkite-agent\builds\buildkite-windows-cpu-autoscaling-group-i-0b3782d1791676daf-1\xgboost\xgboost-ci-windows\src\context.cc:44: No visible GPU is found, setting device to CPU.
  warnings.warn(smsg, UserWarning)
{'Model': 'XGBRegressor', 'RMSE_Train': 7408.723690858731, 'MAE_Train': 5069.22102699311, 'R2_Train': 0.8346486548383104, 'Adj_R2_Train': 0.8342641833036186, 'RMSE_Test': 6794.449488406297, 'MAE_Test': 4951.9730363649105, 'R2_Test': 0.8222965802171944, 'Adj_R2_Test': 0.8167703120181684}
```

Out[54]:

	Model	RMSE_Train	MAE_Train	R2_Train	Adj_R2_Train	RMSE_Test	MAE_Test	R2_Test	Adj_R2_Test
0	LinearRegression	9196.055248	6324.875324	0.745244	0.744652	6753.577695	5219.836166	0.824428	0.818968
1	XGBRegressor	7408.723691	5069.221027	0.834649	0.834264	6794.449488	4951.973036	0.822297	0.816770

Results:

LinearRegression:

- Before Polynominal Features: MAE: 4495.405879 R2: 0.860075
- After Polynominal Features: MAE: 5219.8363 R2: 0.824428

-> Polynominal Features did not improve the model and will not be used

Skewness

In [55]:
```
#pd.set_option('display.max_rows', None)
num_columns_float = df_nans_handeled_cat.select_dtypes(include='number').columns
skew = df_nans_handeled_cat[num_columns_float].skew().sort_values(ascending=False)
skew[skew >= 3]
```

Out[55]:
```
Assortment_b            10.995450
StoreType_b              7.912346
StateHoliday_c           7.611515
StateHoliday_b           5.003762
Customers_Lag_1_MA_8     3.079217
Customers_Lag_2_MA_8     3.073890
Customers_Lag_3_MA_8     3.068401
Customers_Lag_1_MA_6     3.065142
Customers_Lag_4_MA_8     3.063038
Customers_Lag_2_MA_6     3.060226
Customers_Lag_5_MA_8     3.059181
Customers_Lag_3_MA_6     3.055243
Customers_Lag_6_MA_8     3.054436
Customers_Lag_7_MA_8     3.050417
Customers_Lag_1_MA_4     3.049608
Customers_Lag_4_MA_6     3.049238
Customers_Lag_8_MA_8     3.046789
Customers_Lag_2_MA_4     3.046094
Customers_Lag_5_MA_6     3.044501
Customers_Lag_3_MA_4     3.040599
Customers_Lag_6_MA_6     3.038744
Customers_Lag_4_MA_4     3.034761
Customers_Lag_7_MA_6     3.034684
Customers_Lag_5_MA_4     3.031263
Customers_Lag_8_MA_6     3.030890
Customers_Lag_6_MA_4     3.024336
Customers_Lag_7_MA_4     3.018899
Customers_Lag_8_MA_4     3.014113
dtype: float64
```

In [56]:
```python
# Before
testModelsTestSplit8W(df_nans_handeled_cat, None)
```

```
{'Model': 'LinearRegression', 'RMSE_Train': 9348.624262728845, 'MAE_Train': 6448.18666177155, 'R2_Train': 0.7367209231767318, 'Adj_R2_Train': 0.7362865518161379, 'RMSE_Test': 6029.131809472689, 'MAE_Test': 4495.405879127806, 'R2_Test': 0.8600745236580398, 'Adj_R2_Test': 0.8570124514786959}
c:\Users\Chris\anaconda3\lib\site-packages\xgboost\core.py:160: UserWarning: [22:15:22] WARNING: C:\buildkite-agent\builds\buildkite-windows-cpu-autoscaling-group-i-0b3782d1791676daf-1\xgboost\xgboost-ci-windows\src\context.cc:44: No visible GPU is found, setting device to CPU.
  warnings.warn(smsg, UserWarning)
{'Model': 'XGBRegressor', 'RMSE_Train': 7410.980873242441, 'MAE_Train': 5030.000189809052, 'R2_Train': 0.8345478857952568, 'Adj_R2_Train': 0.8342749143885373, 'RMSE_Test': 6058.148984876124, 'MAE_Test': 4515.474892052621, 'R2_Test': 0.858724407972136, 'Adj_R2_Test': 0.8556327904105729}
```

Out[56]:

	Model	RMSE_Train	MAE_Train	R2_Train	Adj_R2_Train	RMSE_Test	MAE_Test	R2_Test	Adj_R2_Test
0	LinearRegression	9348.624263	6448.186662	0.736721	0.736287	6029.131809	4495.405879	0.860075	0.857012
1	XGBRegressor	7410.980873	5030.000190	0.834548	0.834275	6058.148985	4515.474892	0.858724	0.855633

In [57]:
```python
# Log transformation
df_nans_handeled_cat_log = df_nans_handeled_cat.copy()
for_log_transform = skew[skew >= 3].index
df_nans_handeled_cat_log[for_log_transform] = np.log(df_nans_handeled_cat_log[for_log_transform]+1)
testModelsTestSplit8W(df_nans_handeled_cat_log, None)
```

```
{'Model': 'LinearRegression', 'RMSE_Train': 9011.436940430183, 'MAE_Train': 6141.00339989779, 'R2_Train': 0.7553703847711067, 'Adj_R2_Train': 0.7549667822513368, 'RMSE_Test': 6074.646609651307, 'MAE_Test': 4512.055024525082, 'R2_Test': 0.8579539134781295, 'Adj_R2_Test': 0.8548454347286248}
c:\Users\Chris\anaconda3\lib\site-packages\xgboost\core.py:160: UserWarning: [22:16:00] WARNING: C:\buildkite-agent\builds\buildkite-windows-cpu-autoscaling-group-i-0b3782d1791676daf-1\xgboost\xgboost-ci-windows\src\context.cc:44: No visible GPU is found, setting device to CPU.
  warnings.warn(smsg, UserWarning)
{'Model': 'XGBRegressor', 'RMSE_Train': 7410.980873242441, 'MAE_Train': 5030.000189809052, 'R2_Train': 0.8345478857952568, 'Adj_R2_Train': 0.8342749143885373, 'RMSE_Test': 6058.148984876124, 'MAE_Test': 4515.474892052621, 'R2_Test': 0.858724407972136, 'Adj_R2_Test': 0.8556327904105729}
```

Out[57]:

	Model	RMSE_Train	MAE_Train	R2_Train	Adj_R2_Train	RMSE_Test	MAE_Test	R2_Test	Adj_R2_Test
0	LinearRegression	9011.436940	6141.00340	0.755370	0.754967	6074.646610	4512.055025	0.857954	0.854845
1	XGBRegressor	7410.980873	5030.00019	0.834548	0.834275	6058.148985	4515.474892	0.858724	0.855633

In [58]:
```python
# use PowerTransformer to transform the data
df_nans_handeled_cat_power = df_nans_handeled_cat.copy()
for_log_transform = skew[skew >= 3].index
from sklearn.preprocessing import PowerTransformer
pt = PowerTransformer()
df_nans_handeled_cat_power[for_log_transform] = pt.fit_transform(df_nans_handeled_cat_power[for_log_transform])
testModelsTestSplit8W(df_nans_handeled_cat_power, None)
```

```
{'Model': 'LinearRegression', 'RMSE_Train': 9080.95770607921, 'MAE_Train': 6230.562682192093, 'R2_Train': 0.7515813243113636, 'Adj_R2_Train': 0.7511714704047873, 'RMSE_Test': 5959.227147139775, 'MAE_Test': 4414.332558622488, 'R2_Test': 0.8633004401241766, 'Adj_R2_Test': 0.860308962587939}
c:\Users\Chris\anaconda3\lib\site-packages\xgboost\core.py:160: UserWarning: [22:16:29] WARNING: C:\buildkite-agent\builds\buildkite-windows-cpu-autoscaling-group-i-0b3782d1791676daf-1\xgboost\xgboost-ci-windows\src\context.cc:44: No visible GPU is found, setting device to CPU.
  warnings.warn(smsg, UserWarning)
{'Model': 'XGBRegressor', 'RMSE_Train': 7410.980873242441, 'MAE_Train': 5030.000189809052, 'R2_Train': 0.8345478857952568, 'Adj_R2_Train': 0.8342749143885373, 'RMSE_Test': 6058.148984876124, 'MAE_Test': 4515.474892052621, 'R2_Test': 0.858724407972136, 'Adj_R2_Test': 0.8556327904105729}
```

Out[58]:

	Model	RMSE_Train	MAE_Train	R2_Train	Adj_R2_Train	RMSE_Test	MAE_Test	R2_Test	Adj_R2_Test
0	LinearRegression	9080.957706	6230.562682	0.751581	0.751171	5959.227147	4414.332559	0.863300	0.860309
1	XGBRegressor	7410.980873	5030.000190	0.834548	0.834275	6058.148985	4515.474892	0.858724	0.855633

Results:

- Before: MAE:4793.96383 R2:0.844227
- After Power Transformation skew>=2: MAE:4615.821447 R2: 0.850871
- After Power Transformation skew>=3: MAE:4682.241185 R2: 0.847547
- After log transformation skew>=2: MAE: 4641.626858 R2: 0.851209
- After log transformation skew>=3: MAE: 4528.091135 R2: 0.854112

-> Log transformation with skew>=3 will be used

After adding ma3:

- Before: MAE: 4495.405879 R2: 0.860075
- After log transformation skew>=3: MAE: 4512.055025 R2:0.857954
- After Power transformation skew>=3: MAE: 4414.332559 R2: 0.8633

--> Power transformation with skew>=3 will be used

Feature Scaling

In [60]:
```python
# Before
testModelsTestSplit8W(df_nans_handeled_cat_power, None)
```

```
{'Model': 'LinearRegression', 'RMSE_Train': 9080.95770607921, 'MAE_Train': 6230.562682192093, 'R2_Train': 0.7515813243113636, 'Adj_R2_Train': 0.7511714704047873, 'RMSE_Test': 5959.227147139775, 'MAE_Test': 4414.332558622488, 'R2_Test': 0.8633004401241766, 'Adj_R2_Test': 0.860308962587939}
c:\Users\Chris\anaconda3\lib\site-packages\xgboost\core.py:160: UserWarning: [22:18:01] WARNING: C:\buildkite-agent\builds\buildkite-windows-cpu-autoscaling-group-i-0b3782d1791676daf-1\xgboost\xgboost-ci-windows\src\context.cc:44: No visible GPU is found, setting device to CPU.
  warnings.warn(smsg, UserWarning)
{'Model': 'XGBRegressor', 'RMSE_Train': 7410.980873242441, 'MAE_Train': 5030.000189809052, 'R2_Train': 0.8345478857952568, 'Adj_R2_Train': 0.8342749143885373, 'RMSE_Test': 6058.148984876124, 'MAE_Test': 4515.474892052621, 'R2_Test': 0.858724407972136, 'Adj_R2_Test': 0.8556327904105729}
```

Out[60]:

	Model	RMSE_Train	MAE_Train	R2_Train	Adj_R2_Train	RMSE_Test	MAE_Test	R2_Test	Adj_R2_Test
0	LinearRegression	9080.957706	6230.562682	0.751581	0.751171	5959.227147	4414.332559	0.863300	0.860309
1	XGBRegressor	7410.980873	5030.000190	0.834548	0.834275	6058.148985	4515.474892	0.858724	0.855633

StandardScaler

In [59]:
```python
testModelsTestSplit8W(df_nans_handeled_cat_power, StandardScaler())
```

```
{'Model': 'LinearRegression', 'RMSE_Train': 9081.040878116439, 'MAE_Train': 6229.594563260584, 'R2_Train': 0.7515767737820847, 'Adj_R2_Train': 0.7511669123678111, 'RMSE_Test': 5962.061855840987, 'MAE_Tes
t': 4418.61090177654, 'R2_Test': 0.863170357619569, 'Adj_R2_Test': 0.8601760334107396}
c:\Users\Chris\anaconda3\lib\site-packages\xgboost\core.py:160: UserWarning: [22:17:50] WARNING: C:\buildkite-agent\builds\buildkite-windows-cpu-autoscaling-group-i-0b3782d1791676daf-1\xgboost\xgboost-ci-w
indows\src\context.cc:44: No visible GPU is found, setting device to CPU.
  warnings.warn(smsg, UserWarning)
{'Model': 'XGBRegressor', 'RMSE_Train': 7410.980873242441, 'MAE_Train': 5030.000189809052, 'R2_Train': 0.8345478857952568, 'Adj_R2_Train': 0.8342749143885373, 'RMSE_Test': 6058.148984876124, 'MAE_Test': 45
15.474892052621, 'R2_Test': 0.858724407972136, 'Adj_R2_Test': 0.8556327904105729}
```

Out[59]:

	Model	RMSE_Train	MAE_Train	R2_Train	Adj_R2_Train	RMSE_Test	MAE_Test	R2_Test	Adj_R2_Test
0	LinearRegression	9081.040878	6229.594563	0.751577	0.751167	5962.061856	4418.610902	0.863170	0.860176
1	XGBRegressor	7410.980873	5030.000190	0.834548	0.834275	6058.148985	4515.474892	0.858724	0.855633

MinMaxScaler

In [61]:
```python
testModelsTestSplit8W(df_nans_handeled_cat_power, MinMaxScaler())
```

```
{'Model': 'LinearRegression', 'RMSE_Train': 9081.085750405848, 'MAE_Train': 6227.987120558813, 'R2_Train': 0.751574318700868, 'Adj_R2_Train': 0.7511644532360752, 'RMSE_Test': 5941.910085649842, 'MAE_Test':
4398.551737668162, 'R2_Test': 0.8640937628566556, 'Adj_R2_Test': 0.8611196460722401}
c:\Users\Chris\anaconda3\lib\site-packages\xgboost\core.py:160: UserWarning: [22:18:25] WARNING: C:\buildkite-agent\builds\buildkite-windows-cpu-autoscaling-group-i-0b3782d1791676daf-1\xgboost\xgboost-ci-w
indows\src\context.cc:44: No visible GPU is found, setting device to CPU.
  warnings.warn(smsg, UserWarning)
{'Model': 'XGBRegressor', 'RMSE_Train': 7410.980873242441, 'MAE_Train': 5030.000189809052, 'R2_Train': 0.8345478857952568, 'Adj_R2_Train': 0.8342749143885373, 'RMSE_Test': 6058.148984876124, 'MAE_Test': 45
15.474892052621, 'R2_Test': 0.858724407972136, 'Adj_R2_Test': 0.8556327904105729}
```

Out[61]:

	Model	RMSE_Train	MAE_Train	R2_Train	Adj_R2_Train	RMSE_Test	MAE_Test	R2_Test	Adj_R2_Test
0	LinearRegression	9081.085750	6227.987121	0.751574	0.751164	5941.910086	4398.551738	0.864094	0.861120
1	XGBRegressor	7410.980873	5030.000190	0.834548	0.834275	6058.148985	4515.474892	0.858724	0.855633

RobustScaler

In [62]:
```python
testModelsTestSplit8W(df_nans_handeled_cat_power, RobustScaler())
```

```
{'Model': 'LinearRegression', 'RMSE_Train': 9081.139632074473, 'MAE_Train': 6228.870536176268, 'R2_Train': 0.7515713706767402, 'Adj_R2_Train': 0.7511615003481455, 'RMSE_Test': 5945.909267236711, 'MAE_Tes
t': 4398.59384809417, 'R2_Test': 0.8639107588669345, 'Adj_R2_Test': 0.8609326372976843}
c:\Users\Chris\anaconda3\lib\site-packages\xgboost\core.py:160: UserWarning: [22:18:42] WARNING: C:\buildkite-agent\builds\buildkite-windows-cpu-autoscaling-group-i-0b3782d1791676daf-1\xgboost\xgboost-ci-w
indows\src\context.cc:44: No visible GPU is found, setting device to CPU.
  warnings.warn(smsg, UserWarning)
{'Model': 'XGBRegressor', 'RMSE_Train': 7410.980873242441, 'MAE_Train': 5030.000189809052, 'R2_Train': 0.8345478857952568, 'Adj_R2_Train': 0.8342749143885373, 'RMSE_Test': 6058.148984876124, 'MAE_Test': 45
15.474892052621, 'R2_Test': 0.858724407972136, 'Adj_R2_Test': 0.8556327904105729}
```

Out[62]:

	Model	RMSE_Train	MAE_Train	R2_Train	Adj_R2_Train	RMSE_Test	MAE_Test	R2_Test	Adj_R2_Test
0	LinearRegression	9081.139632	6228.870536	0.751571	0.751162	5945.909267	4398.593848	0.863911	0.860933
1	XGBRegressor	7410.980873	5030.000190	0.834548	0.834275	6058.148985	4515.474892	0.858724	0.855633

Result: StandardScaler, MinMaxScaler, RobustScaler have just a small effect on the model performance.

- Before at LinearRegression: MAE: 4414.332559 R2: 0.8633
- After MinMaxScaler at LinearRegression: MAE: 4398.551738 R2: 0.864094

-> MinMaxScaler will be used

In [ ]:
```python
# Save df as csv
df_nans_handeled_cat_power.to_csv('df_nans_handeled_cat_power.csv', index=False)
```

In [63]:
```python
df_nans_handeled_cat_power
```

Out[63]:

	Store	CW	Month	Year	Open	Promo	IsPromo	IsStateHoliday	SchoolHoliday	IsSchoolHoliday	NumStateHoliday	CompetitionDistance	IsCompetition	Promo2	Promo2Member	Promo2Active	StateHoliday_0	S
15	1	16	4	2013	6	0	0	0	0	0	0	1270.0	1	0	0	0	1.0	
16	1	17	4	2013	6	5	1	0	0	0	0	1270.0	1	0	0	0	1.0	
17	1	18	5	2013	5	5	1	1	0	0	1	1270.0	1	0	0	0	0.0	
18	1	19	5	2013	5	0	0	1	0	0	1	1270.0	1	0	0	0	0.0	
19	1	20	5	2013	6	5	1	0	0	0	0	1270.0	1	0	0	0	1.0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
150512	1115	19	5	2015	6	5	1	0	0	0	0	5350.0	0	1	1	0	1.0	
150513	1115	20	5	2015	5	0	0	1	0	0	1	5350.0	0	1	1	0	0.0	
150514	1115	21	5	2015	6	5	1	0	0	0	0	5350.0	0	1	1	0	1.0	
150515	1115	22	5	2015	5	0	0	1	0	0	1	5350.0	0	1	1	0	0.0	
150516	1115	23	6	2015	5	5	1	1	0	0	1	5350.0	0	1	1	1	0.0	

124880 rows × 192 columns

Feature reduction

In [47]:
```python
train_data = []
test_data = []
# Group by store and split into training and test data
amount_test_weeks = 8
for store_id, group in df_nans_handeled_cat_power.groupby('Store'):
        train_data.append(group[: -amount_test_weeks])
        test_data.append(group[-amount_test_weeks:])
# Combine the list entries to one dataframe
train_df = pd.concat(train_data)
test_df = pd.concat(test_data)
X_train = train_df.drop(columns=['Future_Sales'])
y_train = train_df['Future_Sales']
X_test = test_df.drop(columns=['Future_Sales'])
y_test = test_df['Future_Sales']

from sklearn.feature_selection import SequentialFeatureSelector
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import TimeSeriesSplit

model = LinearRegression(n_jobs=-1)
tscv = TimeSeriesSplit(n_splits=3)
sfs = SequentialFeatureSelector(model, n_features_to_select=40, direction='forward', cv=tscv)

# perform feature selection
sfs.fit(X_train, y_train)

# get the selected feature indices
selected_features_boolean = sfs.get_support()

# extract selected column names
selected_columns = X_train.columns[selected_features_boolean]
```

```
print("Ausgewählte Spaltennamen:", selected_columns)
```

```
Ausgewählte Spaltennamen: Index(['CW', 'Month', 'Open', 'Promo', 'IsPromo', 'IsSchoolHoliday',
       'StateHoliday_a', 'StateHoliday_b', 'Assortment_b', 'Assortment_c',
       'Sales_Lag_1', 'Sales_Lag_1_MA_6', 'Sales_Lag_1_MA_8',
       'Sales_Lag_2_MA_8', 'Sales_Lag_3', 'Sales_Lag_3_MA_8',
       'Sales_Lag_4_MA_4', 'Sales_Lag_5_MA_8', 'Sales_Lag_7_MA_8',
       'Sales_Lag_8_MA_8', 'SalesPerCustomer_Lag_3_MA_6',
       'SalesPerCustomer_Lag_4_MA_6', 'SalesPerCustomer_Lag_7',
       'SalesPerCustomer_Lag_8', 'SalesPerOpenDay_Lag_1',
       'SalesPerOpenDay_Lag_4_MA_6', 'SalesPerOpenDay_Lag_5_MA_8',
       'SalesPerOpenDay_Lag_7', 'Customers_Lag_1_MA_4', 'Customers_Lag_1_MA_6',
       'Customers_Lag_3_MA_4', 'Customers_Lag_4', 'Customers_Lag_4_MA_8',
       'Customers_Lag_6', 'Customers_Lag_7_MA_8', 'Customers_Lag_8_MA_8',
       'CustomersPerOpenDay_Lag_6', 'CustomersPerOpenDay_Lag_6_MA_4',
       'CustomersPerOpenDay_Lag_7_MA_4', 'CustomersPerOpenDay_Lag_8'],
      dtype='object')
```

In [53]:

```
train_data = []
test_data = []
# Group by store and split into training and test data
amount_test_weeks = 8
df_to_reduce = df_nans_handeled_cat_power[['Store', 'Future_Sales', 'CW', 'Month', 'Open', 'Promo', 'IsPromo', 'IsSchoolHoliday',
       'StateHoliday_a', 'StateHoliday_b', 'Assortment_b', 'Assortment_c',
       'Sales_Lag_1', 'Sales_Lag_1_MA_6', 'Sales_Lag_1_MA_8',
       'Sales_Lag_2_MA_8', 'Sales_Lag_3', 'Sales_Lag_3_MA_8',
       'Sales_Lag_4_MA_4', 'Sales_Lag_5_MA_8', 'Sales_Lag_7_MA_8',
       'Sales_Lag_8_MA_8', 'SalesPerCustomer_Lag_3_MA_6',
       'SalesPerCustomer_Lag_4_MA_6', 'SalesPerCustomer_Lag_7',
       'SalesPerCustomer_Lag_8', 'SalesPerOpenDay_Lag_1',
       'SalesPerOpenDay_Lag_4_MA_6', 'SalesPerOpenDay_Lag_5_MA_8',
       'SalesPerOpenDay_Lag_7', 'Customers_Lag_1_MA_4', 'Customers_Lag_1_MA_6',
       'Customers_Lag_3_MA_4', 'Customers_Lag_4', 'Customers_Lag_4_MA_8',
       'Customers_Lag_6', 'Customers_Lag_7_MA_8', 'Customers_Lag_8_MA_8',
       'CustomersPerOpenDay_Lag_6', 'CustomersPerOpenDay_Lag_6_MA_4',
       'CustomersPerOpenDay_Lag_7_MA_4', 'CustomersPerOpenDay_Lag_8']]
for store_id, group in df_to_reduce.groupby('Store'):
        train_data.append(group[: -amount_test_weeks])
        test_data.append(group[-amount_test_weeks:])
# Combine the list entries to one dataframe
train_df = pd.concat(train_data)
test_df = pd.concat(test_data)
X_train = train_df.drop(columns=['Future_Sales'])
y_train = train_df['Future_Sales']
X_test = test_df.drop(columns=['Future_Sales'])
y_test = test_df['Future_Sales']

from sklearn.feature_selection import SequentialFeatureSelector
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import TimeSeriesSplit

model = LinearRegression(n_jobs=-1)
tscv = TimeSeriesSplit(n_splits=3)
sfs = SequentialFeatureSelector(model, n_features_to_select=20, direction='forward', cv=tscv)

# perform feature selection
sfs.fit(X_train, y_train)

# get the selected feature indices
selected_features_boolean = sfs.get_support()

# extract selected column names
selected_columns = X_train.columns[selected_features_boolean]

print("Ausgewählte Spaltennamen:", selected_columns)
```

```
Ausgewählte Spaltennamen: Index(['Open', 'Promo', 'IsPromo', 'StateHoliday_b', 'Sales_Lag_1',
       'Sales_Lag_1_MA_6', 'Sales_Lag_1_MA_8', 'Sales_Lag_2_MA_8',
       'Sales_Lag_3_MA_8', 'Sales_Lag_5_MA_8', 'Sales_Lag_8_MA_8',
       'SalesPerCustomer_Lag_3_MA_6', 'SalesPerCustomer_Lag_8',
       'SalesPerOpenDay_Lag_1', 'SalesPerOpenDay_Lag_4_MA_6',
       'Customers_Lag_1_MA_6', 'Customers_Lag_3_MA_4', 'Customers_Lag_6',
       'Customers_Lag_8_MA_8', 'CustomersPerOpenDay_Lag_6'],
      dtype='object')
```

In [63]:

```
# with 40 selected features
df_test = df_nans_handeled_cat_power[['Store', 'Future_Sales', 'CW', 'Month', 'Open', 'Promo', 'IsPromo', 'IsSchoolHoliday',
       'StateHoliday_a', 'StateHoliday_b', 'Assortment_b', 'Assortment_c',
       'Sales_Lag_1', 'Sales_Lag_1_MA_6', 'Sales_Lag_1_MA_8',
       'Sales_Lag_2_MA_8', 'Sales_Lag_3', 'Sales_Lag_3_MA_8',
       'Sales_Lag_4_MA_4', 'Sales_Lag_5_MA_8', 'Sales_Lag_7_MA_8',
       'Sales_Lag_8_MA_8', 'SalesPerCustomer_Lag_3_MA_6',
       'SalesPerCustomer_Lag_4_MA_6', 'SalesPerCustomer_Lag_7',
       'SalesPerCustomer_Lag_8', 'SalesPerOpenDay_Lag_1',
       'SalesPerOpenDay_Lag_4_MA_6', 'SalesPerOpenDay_Lag_5_MA_8',
       'SalesPerOpenDay_Lag_7', 'Customers_Lag_1_MA_4', 'Customers_Lag_1_MA_6',
       'Customers_Lag_3_MA_4', 'Customers_Lag_4', 'Customers_Lag_4_MA_8',
       'Customers_Lag_6', 'Customers_Lag_7_MA_8', 'Customers_Lag_8_MA_8',
       'CustomersPerOpenDay_Lag_6', 'CustomersPerOpenDay_Lag_6_MA_4',
       'CustomersPerOpenDay_Lag_7_MA_4', 'CustomersPerOpenDay_Lag_8']]
testModelsTestSplit8W(df_test, None)
```

```
{'Model': 'LinearRegression', 'RMSE_Train': 9243.960853576737, 'MAE_Train': 6346.849079059969, 'R2_Train': 0.7425830551488228, 'Adj_R2_Train': 0.7424920072120149, 'RMSE_Test': 6076.866293181645, 'MAE_Tes
t': 4604.182223657593, 'R2_Test': 0.8578500868741831, 'Adj_R2_Test': 0.8571936162233429}
/usr/local/lib/python3.10/dist-packages/xgboost/core.py:160: UserWarning: [20:25:06] WARNING: /workspace/src/context.cc:44: No visible GPU is found, setting device to CPU.
  warnings.warn(smsg, UserWarning)
{'Model': 'XGBRegressor', 'RMSE_Train': 7586.747425730487, 'MAE_Train': 5128.276979852914, 'R2_Train': 0.8266067494562147, 'Adj_R2_Train': 0.8265454205575768, 'RMSE_Test': 6139.361351649675, 'MAE_Test': 46
97.845359576443, 'R2_Test': 0.8549112868641222, 'Adj_R2_Test': 0.854241244372731}
{'Model': 'RandomForestRegressor', 'RMSE_Train': 6833.118635588801, 'MAE_Train': 4610.654046070799, 'R2_Train': 0.8593438071493607, 'Adj_R2_Train': 0.859294057292506, 'RMSE_Test': 6633.261060700181, 'MAE_T
est': 5053.971024294538, 'R2_Test': 0.8306280776890549, 'Adj_R2_Test': 0.8298458915193377}
```

Out[63]:

	Model	RMSE_Train	MAE_Train	R2_Train	Adj_R2_Train	RMSE_Test	MAE_Test	R2_Test	Adj_R2_Test
0	LinearRegression	9243.960854	6346.849079	0.742583	0.742492	6076.866293	4604.182224	0.857850	0.857194
1	XGBRegressor	7586.747426	5128.276979	0.826607	0.826545	6139.361352	4697.845360	0.854911	0.854241
2	RandomForestRegressor	6833.118636	4610.654046	0.859344	0.859294	6633.261061	5053.971024	0.830628	0.829846

In [49]:

```
# with 20 selected features

df_test = df_nans_handeled_cat_power[['Store', 'Future_Sales', 'Open', 'Promo', 'IsPromo', 'StateHoliday_b', 'Sales_Lag_1',
       'Sales_Lag_1_MA_6', 'Sales_Lag_1_MA_8', 'Sales_Lag_2_MA_8',
       'Sales_Lag_3_MA_8', 'Sales_Lag_5_MA_8', 'Sales_Lag_8_MA_8',
       'SalesPerCustomer_Lag_3_MA_6', 'SalesPerCustomer_Lag_8',
       'SalesPerOpenDay_Lag_1', 'SalesPerOpenDay_Lag_4_MA_6',
       'Customers_Lag_1_MA_6', 'Customers_Lag_3_MA_4', 'Customers_Lag_6',
       'Customers_Lag_8_MA_8', 'CustomersPerOpenDay_Lag_6']]
testModelsTestSplit8W(df_test, None)
```

```
{'Model': 'LinearRegression', 'RMSE_Train': 9536.329179581322, 'MAE_Train': 6673.629827187907, 'R2_Train': 0.7260423678114407, 'Adj_R2_Train': 0.7259927455109356, 'RMSE_Test': 5775.47628316968, 'MAE_Test':
4295.672175582573, 'R2_Test': 0.8716006437113271, 'Adj_R2_Test': 0.87129761084 07874}
c:\Users\Chris\anaconda3\lib\site-packages\xgboost\core.py:160: UserWarning: [22:09:47] WARNING: C:\buildkite-agent\builds\buildkite-windows-cpu-autoscaling-group-i-0b3782d1791676daf-1\xgboost\xgboost-ci-w
indows\src\context.cc:44: No visible GPU is found, setting device to CPU.
  warnings.warn(smsg, UserWarning)
{'Model': 'XGBRegressor', 'RMSE_Train': 9457.068947317517, 'MAE_Train': 6669.648039198498, 'R2_Train': 0.7305773851169427, 'Adj_R2_Train': 0.7305285842499919, 'RMSE_Test': 6298.190214583256, 'MAE_Test': 49
20.265377574972, 'R2_Test': 0.8473071214064962, 'Adj_R2_Test': 0.8469467538575567}
c:\Users\Chris\anaconda3\lib\site-packages\scikeras\wrappers.py:915: UserWarning: ``build_fn`` will be renamed to ``model`` in a future release, at which point use of ``build_fn`` will raise an Error inste
ad.
  X, y = self._initialize(X, y)
```

{'Model': 'NeuralNetwork', 'RMSE_Train': 12397.468610666763, 'MAE_Train': 9263.2136661244, 'R2_Train': 0.5369936247054163, 'Adj_R2_Train': 0.5369097597613843, 'RMSE_Test': 11164.801701721253, 'MAE_Test': 9001.246323877172, 'R2_Test': 0.5201679368692144, 'Adj_R2_Test': 0.5190354943736258}

```
c:\Users\Chris\anaconda3\lib\site-packages\sklearn\neural_network\_multilayer_perceptron.py:691: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (200) reached and the optimization hasn't converged yet.
  warnings.warn(
```

{'Model': 'MLPRegressor', 'RMSE_Train': 8739.366559683718, 'MAE_Train': 5797.459515852285, 'R2_Train': 0.7699189517258402, 'Adj_R2_Train': 0.7698772768477695, 'RMSE_Test': 6647.838726860592, 'MAE_Test': 5149.058839719626, 'R2_Test': 0.8298828151753054, 'Adj_R2_Test': 0.8294813248537366}

Out[49]:

	Model	RMSE_Train	MAE_Train	R2_Train	Adj_R2_Train	RMSE_Test	MAE_Test	R2_Test	Adj_R2_Test
0	LinearRegression	9536.329180	6673.629827	0.726042	0.725993	5775.476283	4295.672176	0.871601	0.871298
1	XGBRegressor	9457.068947	6669.648039	0.730577	0.730529	6298.190215	4920.265378	0.847307	0.846947
2	NeuralNetwork	12397.468611	9263.213666	0.536994	0.536910	11164.801702	9001.246324	0.520168	0.519035
3	MLPRegressor	8739.366560	5797.459516	0.769919	0.769877	6647.838727	5149.058840	0.829883	0.829481

Result: LinearRegresion without scaler and with all 192 features: MAE: 4414 R2:0.86

- With 40 features MAE: 4604 R2:0.857
- With 20 features MAE: 4295 R2: 0.871

-> The Top 20 will be used