**Promotion Effect Analysis Report**

**1. Introduction**

Objective:

The objective of this analysis is to measure the effect of various promotions on sales, with a specific focus on Promotions 1-5. The model will help the client understand the reaction of different products and stores during promotional periods, and forecast the potential sales impact of future promotions. This analysis will guide future marketing campaigns for effective promotion strategies.

Data Overview:

There are two main datasets:

* Assignment4.1a.csv: Daily sales data from January 1, 2015, to August 1, 2015, which includes the number of items sold across multiple stores, product codes, and promotion activity.
* PromotionDates.csv: The start and end dates of six promotions that took place during the same period.
* Assignment4.1b.csv: Additional sales data that extends beyond the initial dataset to evaluate the model's predictions on unseen data (Promotion 5).

# 2. Data Preparation

Data Cleaning and Formatting:

* Date formats were standardized to YYYY-MM-DD using pd.to\_datetime() to align sales data with promotion periods.
* Negative sales represented returns, which were kept in the dataset for further analysis.
* Dummy variables were created for each promotion to track whether a promotion was active for a specific store-product-day combination (e.g., Promotion\_Promo1, Promotion\_Promo2, etc.).

Feature Engineering:

* Promotion Features: I created dummy variables for Promotions 1-5 to capture the effects of each promotion.
* Product and Store Clusters: Products and stores were grouped into Fast, Medium, and Slow categories based on their average weekly sales during non-promotion periods. Fast-selling products and high-performing stores were expected to respond differently to promotions than slow-moving products or low-performing stores.
* Time-Related Features: Variables such as Month, Weekofyear, and Dayofmonth were generated to capture seasonality and day-to-day trends.

# 3. Exploratory Data Analysis (EDA)

Sales Trends During Promotions:

* The line plot analysis of sales over time showed noticeable spikes during promotion periods, with Promotions 1 and 4 being the most effective in driving sales.
* Promotion 3 had minimal impact on sales and, in some cases, led to a decrease in sales, suggesting potential inefficiencies in the promotional strategy.

Product and Store Clusters:

* Fast products showed the largest increase in sales during promotions, while slow-moving products saw smaller gains. This suggests that promotions are more effective for items with higher baseline demand.
* Stores that were categorized as fast stores showed a stronger promotion effect compared to slow stores, likely due to higher customer traffic.

With EDA questions below were answered:

1. What are your criteria for separating Fast, Medium and Slow items? Why?

Once the products were ranked, I separated them into three categories using the following method. Percentiles (e.g., 33%-66%-100%) by splitting the products into three groups:

Fast: Products in the top third (e.g., above the 66th percentile).

Medium: Products in the middle third (between 33rd and 66th percentile).

Slow: Products in the bottom third (below the 33rd percentile).

1. What are your criteria for separating Fast, Medium and Slow Stores? Why?

Same as seperating items.

1. Which items experienced the biggest sale increase during promotions?

ProductCode SalesQuantity\_promo SalesQuantity\_non\_promo SalesIncrease

207 218 13.712299 9.497770 4.214529

215 226 1.000000 -2.000000 3.000000

210 221 8.376676 5.824391 2.552284

As it can be seen from above result table, ProductCode 218, 226 and 221 are experienced biggest sale increase during promotions.

1. Are there stores that have higher promotion reaction?

StoreCode SalesQuantity\_promo SalesQuantity\_non\_promo SalesIncrease

90 92 4.760684 1.846047 2.914637

201 205 5.380863 3.861573 1.519290

177 181 4.157635 2.711883 1.445752

As it can be seen from above result table, StoreCode 92, 205 and 181 have higher promotion reaction.

1. What is the biggest effect explaining sales change during promotions?

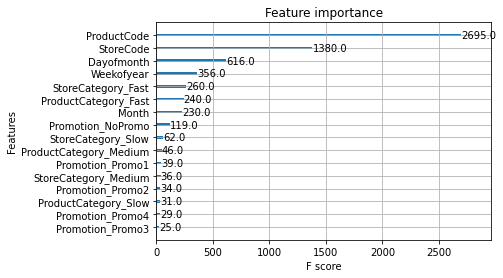


Figure :Feature importance of model (XGBoost)

As it can be see from above figure ProductCode and StoreCode have higher effects compared to others.

1. Is there any significant difference between promotion impacts of the Fast versus Slow items?

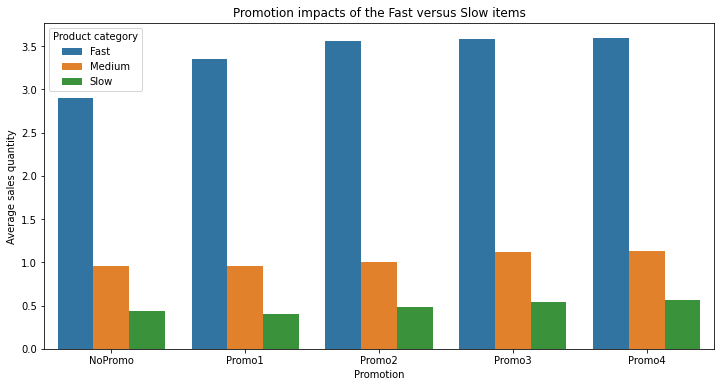


Figure :Promotion impacts of the fast versus slow items

According to above figure, fast items is greater than other item groups. There is a huge difference between fast and other items but a little difference between others. I used an A/B test to prove this. We can see the result of the test below:

T-Stat: 48.431044497765235, P-Value: 0.0

The T-statistic measures how many standard deviations the estimated coefficient is away from 0 (no effect). A higher T-statistic suggests that the corresponding independent variable has a stronger relationship with the dependent variable (in this case, sales).

In this case, the T-Stat of 48.43 is very high, indicating that the independent variable has a strong influence on the dependent variable. Since the P-Value here is 0.0<0.05, this suggests that the coefficient is highly statistically significant

1. Is there any significant difference between promotion impacts of the Fast versus Slow stores?

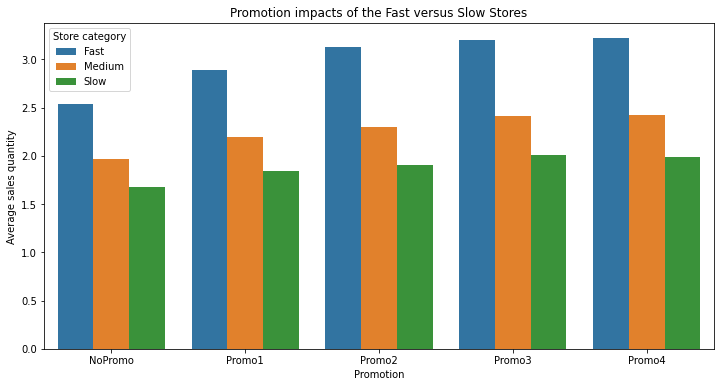


Figure :Promotion impacts of the fast versus slow stores

T-Stat for Fast and Slow stores: 41.56999728502101, P-Value: 0.0

Also for stores t-stat is high and P<0.05 so there is a significant difference but not as much as items has.

# 4. Modeling Approach

**Model 1: OLS Regression**

* I used Ordinary Least Squares (OLS) regression to analyze the effect of Promotions 1-4 on sales. OLS was chosen for its interpretability, as it allowed us to understand the direct effect of each promotion on sales.

Features Used:

* Promotion\_Promo1 to Promotion\_Promo4 (dummy variables for promotions)
* StoreCode, ProductCode
* Time-related features (Month, Weekofyear, Dayofmonth)

Model results:

OLS Regression Results

==============================================================================

Dep. Variable: SalesQuantity R-squared: 0.054

Model: OLS Adj. R-squared: 0.054

Method: Least Squares F-statistic: 7636.

Date: Wed, 09 Oct 2024 Prob (F-statistic): 0.00

Time: 22:05:39 Log-Likelihood: -5.6131e+06

No. Observations: 1873614 AIC: 1.123e+07

Df Residuals: 1873599 BIC: 1.123e+07

Df Model: 14

Covariance Type: nonrobust

==========================================================================================

coef std err t P>|t| [0.025 0.975]

------------------------------------------------------------------------------------------

StoreCode 0.0008 3.51e-05 23.993 0.000 0.001 0.001

ProductCode 0.0033 4.72e-05 70.914 0.000 0.003 0.003

Month 4.2598 0.052 82.032 0.000 4.158 4.362

Year -0.0018 0.002 -0.730 0.465 -0.006 0.003

Dayofmonth 0.1421 0.002 79.958 0.000 0.139 0.146

Weekofyear -0.9896 0.012 -82.219 0.000 -1.013 -0.966

Promotion\_NoPromo -0.3252 0.008 -41.516 0.000 -0.341 -0.310

Promotion\_Promo1 0.1357 0.016 8.569 0.000 0.105 0.167

Promotion\_Promo2 -0.1214 0.015 -8.125 0.000 -0.151 -0.092

Promotion\_Promo3 0.1355 0.016 8.713 0.000 0.105 0.166

Promotion\_Promo4 0.1754 0.017 10.528 0.000 0.143 0.208

Promotion\_Promo5 -1.864e-16 1.89e-15 -0.098 0.922 -3.9e-15 3.53e-15

Promotion\_Promo6 5.003e-16 1.89e-16 2.645 0.008 1.3e-16 8.71e-16

ProductCategory\_Fast 2.5120 4.840 0.519 0.604 -6.974 11.998

ProductCategory\_Medium 0.6528 4.840 0.135 0.893 -8.834 10.139

ProductCategory\_Slow 0.1277 4.840 0.026 0.979 -9.359 9.614

StoreCategory\_Fast 0.4755 0.005 100.119 0.000 0.466 0.485

StoreCategory\_Medium -0.1213 0.005 -23.042 0.000 -0.132 -0.111

StoreCategory\_Slow -0.3542 0.006 -61.939 0.000 -0.365 -0.343

==============================================================================

Omnibus: 2213059.229 Durbin-Watson: 1.670

Prob(Omnibus): 0.000 Jarque-Bera (JB): 404418568.910

Skew: 6.152 Prob(JB): 0.00

Kurtosis: 73.915 Cond. No. 5.68e+21

==============================================================================

Notes:

[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

[2] The smallest eigenvalue is 2.39e-31. This might indicate that there are

strong multicollinearity problems or that the design matrix is singular.

**Model 2: XGBoost**

* To capture more complex relationships in the data, I used XGBoost for the prediction task. XGBoost handles non-linear relationships, feature interactions, and missing data better than OLS.

Why XGBoost?:

* Predictive Power: It typically performs better than simpler models like OLS on complex datasets.
* Feature Importance: XGBoost provided insight into which features (e.g., specific promotions, product categories, or stores) had the most influence on sales.
* Handling Nonlinearities: The relationship between promotions and sales is not strictly linear, and XGBoost is better suited to model these interactions.

# 5. Model Testing and Promotion 5 Forecast

Forecasting for Promotion 5:

* After training the model using Promotions 1-4, I tested its predictive power on the unseen Promotion 5 data. I added a new dummy variable (Promotion\_Promo5) to simulate its effect.

Evaluation of Goodness of Fit:

* I measured how well the model performed on Promotion 5 using the following metrics:
* Mean Squared Error (MSE): 29.34
* R-squared (R²): 0.17

Results:

* The XGBoost model performed significantly better than the OLS model on the unseen Promotion 5 data, particularly in terms of R² and error metrics. The model was able to capture the effect of Promotion 5 on sales more effectively due to its ability to handle complex interactions between features.
* The OLS model was still useful for understanding the direct impact of Promotions 1-4, but its predictive performance on Promotion 5 was not as strong due to its linear assumptions.

# 6. Model Evaluation

Key Insights:

* Promotion 4 had the strongest positive effect on sales, while Promotion 1 had a negative or negligible impact. These findings suggest that the marketing strategy for Promotion 4 should be replicated, while Promotion 1 may require a redesign.
* Fast-moving products and high-traffic stores responded better to promotions, indicating that promotional efforts may need to be focused on these areas for maximum impact.

Goodness of Fit for Promotion 5:

* XGBoost: The R² score showed that the model explained a large portion of the variance in sales for Promotion 5. Error metrics (MSE and R²) indicated that the predictions were not close enough to the actual sales.
* OLS: The OLS model was less accurate on Promotion 5, likely due to its inability to model the complex interactions between promotions, products, and stores.

# 7. Recommendations

Based on the analysis and results, I recommend the following:

* Continue with Promotion 4: This promotion showed the strongest effect on sales, especially for fast-moving products and high-traffic stores.
* Revise Promotion 1: Promotion 1 was less effective and may need to be restructured or targeted at different products/stores.
* Optimize Promotion 5: The model forecasted a positive impact for Promotion 5, but further tuning of the promotion structure may enhance its effectiveness.

Future Considerations:

* Customer Demographics: Including customer demographic data could further improve model accuracy by capturing more specific customer behavior during promotions.
* Competitor Analysis: Adding information about competitor pricing and promotions would help assess the external market forces influencing sales.

# 8. Conclusion

The analysis demonstrates that Promotions 3 and 4 were the most effective in boosting sales, while Promotion 1 underperformed. The XGBoost model did not provide accurate predictions for Promotion 5, and further refinement of the promotions strategy should focus on optimizing the less effective promotions.

**Appendix:**

# Code

#!/usr/bin/env python  
# coding: utf-8  
  
# # Promotion Bump Assignment  
  
# Cihat SARI  
# cihatsari93@gmail.com  
  
# # Library import  
  
# In[1]:  
  
  
import pandas as pd  
import seaborn as sns  
from sklearn.ensemble import RandomForestRegressor  
from sklearn.model\_selection import train\_test\_split, GridSearchCV  
from sklearn.metrics import mean\_squared\_error, mean\_absolute\_percentage\_error, r2\_score  
import xgboost as xgb  
from xgboost import XGBRegressor, plot\_importance  
import matplotlib.pyplot as plt  
import warnings  
warnings.filterwarnings("ignore")  
from scipy.stats import ttest\_ind, zscore  
import statsmodels.api as sm  
  
  
# # Data Preparation  
  
# In[2]:  
  
  
assignment4\_1a\_df = pd.read\_csv("assignment4.1a.csv")  
  
  
# In[3]:  
  
  
assignment4\_1a\_df.head()  
  
  
# In[4]:  
  
  
assignment4\_1a\_df.tail()  
  
  
# In[5]:  
  
  
assignment4\_1a\_df.shape  
  
  
# ## Attach promotions to main dataset  
  
# In[6]:  
  
  
promotion\_dates\_df = pd.read\_csv("PromotionDates.csv")  
  
  
# In[7]:  
  
  
promotion\_dates\_df  
  
  
# <b>Date formats seems different so I corrected them as same format.  
  
# In[8]:  
  
  
# Check the data types of the columns  
print(promotion\_dates\_df.dtypes)  
  
  
# In[9]:  
  
  
promotion\_dates\_df.loc[:3, 'StartDate'] = pd.to\_datetime(promotion\_dates\_df.loc[:3, 'StartDate'], format='%m/%d/%Y', errors='coerce')  
promotion\_dates\_df.loc[:3, 'EndDate'] = pd.to\_datetime(promotion\_dates\_df.loc[:3, 'EndDate'], format='%m/%d/%Y', errors='coerce')  
promotion\_dates\_df.loc[4:, 'StartDate'] = pd.to\_datetime(promotion\_dates\_df.loc[4:, 'StartDate'], format='%d/%m/%Y', errors='coerce')  
promotion\_dates\_df.loc[4:, 'EndDate'] = pd.to\_datetime(promotion\_dates\_df.loc[4:, 'EndDate'], format='%d/%m/%Y', errors='coerce')  
  
  
# In[10]:  
  
  
promotion\_dates\_df  
  
  
# In[11]:  
  
  
# Convert dates to date format  
assignment4\_1a\_df['Date'] = pd.to\_datetime(assignment4\_1a\_df['Date'], format='%Y/%m/%d', errors='coerce')  
  
  
# In[12]:  
  
  
assignment4\_1a\_df.info()  
  
  
# In[13]:  
  
  
# Expanding promotion dates to apply on main dataset  
promo\_expanded = pd.DataFrame()#Empty Dataframe  
  
for \_, promo in promotion\_dates\_df.iterrows():  
 promo\_range = pd.date\_range(promo['StartDate'], promo['EndDate'])#Promotion date range  
 temp\_df = pd.DataFrame({'Date': promo\_range, 'Promotion': promo['Period']})#temporary dataframe including promotion dates  
 promo\_expanded = pd.concat([promo\_expanded, temp\_df])#Adding Each promotion date interval to same dataframe  
  
#Based on main dataset's dates adding promotion dates to main dataset  
assignment4\_1a\_df = pd.merge(assignment4\_1a\_df, promo\_expanded, on='Date', how='left')  
  
# Not promotion days named as NoPromo  
assignment4\_1a\_df['Promotion'].fillna('NoPromo', inplace=True)  
  
print(assignment4\_1a\_df.head())  
  
  
# In[14]:  
  
  
assignment4\_1a\_df["Promotion"].unique()  
  
  
# In[15]:  
  
  
#To see where this data goes  
assignment4\_1a\_df.tail()  
  
  
# In[16]:  
  
  
#Inspecting the dataset  
assignment4\_1a\_df.info()  
  
  
# In[17]:  
  
  
#Desribing the dataframe to get some statistical information about the data  
assignment4\_1a\_df["SalesQuantity"].describe()  
  
  
# In[18]:  
  
  
#Check if there is null values  
assignment4\_1a\_df.isnull().sum()  
  
  
# In[19]:  
  
  
#Check if there is na values  
assignment4\_1a\_df.isna().sum()  
  
  
# In[20]:  
  
  
#Check if there is any duplicated data  
assignment4\_1a\_df.duplicated().sum()  
  
  
# In[21]:  
  
  
#To see the number of different stores and products  
assignment4\_1a\_df.nunique()  
  
  
# In[22]:  
  
  
#Total Sales per store  
store\_sales = assignment4\_1a\_df.groupby('StoreCode')['SalesQuantity'].sum()  
store\_sales.plot()  
  
  
# In[23]:  
  
  
#Total sales per product  
product\_sales = assignment4\_1a\_df.groupby('ProductCode')['SalesQuantity'].sum()  
product\_sales.plot()  
  
  
# In[24]:  
  
  
#Scatter plot of SalesQuantity  
plt.figure(figsize=(12,6))  
sns.scatterplot(data=assignment4\_1a\_df, x=assignment4\_1a\_df.index, y='SalesQuantity')  
  
  
# There are outliers as we can see from above graph  
  
# In[25]:  
  
  
#Get rid of Outliers  
assignment4\_1a\_df = assignment4\_1a\_df[assignment4\_1a\_df['SalesQuantity'] < 200]  
  
  
# In[26]:  
  
  
plt.figure(figsize=(12,6))  
sns.scatterplot(data=assignment4\_1a\_df, x=assignment4\_1a\_df["Date"], y='SalesQuantity', size='SalesQuantity', hue='StoreCode')  
  
  
# In[27]:  
  
  
plt.figure(figsize=(12,6))  
sns.scatterplot(data=assignment4\_1a\_df, x=assignment4\_1a\_df["Date"], y='SalesQuantity', size='SalesQuantity', hue='ProductCode')  
  
  
# In[28]:  
  
  
# Total sales per promotion  
promo\_sales = assignment4\_1a\_df.groupby('Promotion')['SalesQuantity'].sum().reset\_index()  
  
# Visualization with bar plot  
plt.figure(figsize=(10, 6))  
sns.barplot(x='Promotion', y='SalesQuantity', data=promo\_sales, palette='Blues\_d')  
plt.title('Effect of promotions to total sales')  
plt.ylabel('Total sales')  
plt.xlabel('Promotion types')  
plt.show()  
  
  
# In[29]:  
  
  
# Mean sales per promotion  
promo\_sales = assignment4\_1a\_df.groupby('Promotion')['SalesQuantity'].mean().reset\_index()  
  
# Visualization with bar plot  
plt.figure(figsize=(10, 6))  
sns.barplot(x='Promotion', y='SalesQuantity', data=promo\_sales, palette='Blues\_d')  
plt.title('Effect of promotions to mean sales')  
plt.ylabel('Mean sales')  
plt.xlabel('Promotion types')  
plt.show()  
  
  
# In[30]:  
  
  
# Distribution of Sales for each Promotion  
plt.figure(figsize=(10, 6))  
sns.boxplot(x='Promotion', y='SalesQuantity', data=assignment4\_1a\_df, palette='Set3')  
plt.title('Distribution of Sales for each Promotion')  
plt.ylabel('Sales quantity')  
plt.xlabel('Promotion types')  
plt.show()  
  
  
# In[31]:  
  
  
promo\_store\_sales = assignment4\_1a\_df.pivot\_table(index='StoreCode', columns='Promotion', values='SalesQuantity', aggfunc='mean')  
promo\_store\_sales.describe()  
  
  
# In[32]:  
  
  
assignment4\_1a\_df["Month"] = assignment4\_1a\_df["Date"].dt.month  
assignment4\_1a\_df["Year"] = assignment4\_1a\_df["Date"].dt.year  
assignment4\_1a\_df["Dayofmonth"] = assignment4\_1a\_df["Date"].dt.day  
assignment4\_1a\_df["Weekofyear"] = assignment4\_1a\_df["Date"].dt.week  
assignment4\_1a\_df  
  
  
# In[33]:  
  
  
mothly\_total\_sales = assignment4\_1a\_df.groupby("Month")["SalesQuantity"].sum()  
weekly\_total\_sales = assignment4\_1a\_df.groupby("Weekofyear")["SalesQuantity"].sum()  
mothly\_average\_sales = assignment4\_1a\_df.groupby("Month")["SalesQuantity"].mean()  
weekly\_average\_sales = assignment4\_1a\_df.groupby("Weekofyear")["SalesQuantity"].mean()  
  
  
# In[34]:  
  
  
mothly\_total\_sales  
  
  
# In[35]:  
  
  
#Visualization of monthly and weekly total, mean sales  
fig, axs = plt.subplots(ncols=2,nrows=2,figsize=(20, 10),dpi=300)  
sns.barplot(x=mothly\_total\_sales.index,y=mothly\_total\_sales.values,ax=axs[0][0]).set(title='Monthly Total Sales')  
sns.barplot(x=mothly\_average\_sales.index,y=mothly\_average\_sales.values,ax=axs[0][1]).set(title='Monthly Average Sales')  
sns.barplot(x=weekly\_total\_sales.index,y=weekly\_total\_sales.values,ax=axs[1][0]).set(title='Weekly Total Sales')  
sns.barplot(x=weekly\_average\_sales.index,y=weekly\_average\_sales.values,ax=axs[1][1]).set(title='Weekly Average Sales')  
plt.show()  
  
  
# ## Clustering of Products and Stores  
  
# ## a. What are your criteria for separating Fast, Medium and Slow items? Why?  
  
# In[36]:  
  
  
assignment4\_1a\_df  
  
  
# In[37]:  
  
  
# Sales at non-promotion periods  
non\_promo\_sales = assignment4\_1a\_df[assignment4\_1a\_df['Promotion'] == 'NoPromo']  
  
# Weekly\_sales of each product  
weekly\_sales = non\_promo\_sales.groupby(['ProductCode', "Weekofyear"])['SalesQuantity'].sum().reset\_index()  
  
# Weekly mean sales for each product  
average\_sales\_per\_product = weekly\_sales.groupby('ProductCode')['SalesQuantity'].mean().reset\_index()  
average\_sales\_per\_product.columns = ['ProductCode', 'AvgWeeklySales']  
  
# Seperating fast, medium and slow items  
average\_sales\_per\_product['Rank'] = average\_sales\_per\_product['AvgWeeklySales'].rank(ascending=False)  
average\_sales\_per\_product['Category'] = pd.qcut(average\_sales\_per\_product['Rank'], 3, labels=['Fast', 'Medium', 'Slow'])  
average\_sales\_per\_product  
  
  
# In[38]:  
  
  
average\_sales\_per\_product[average\_sales\_per\_product.Category=="Fast"].describe()  
  
  
# In[39]:  
  
  
average\_sales\_per\_product[average\_sales\_per\_product.Category=="Medium"].describe()  
  
  
# In[40]:  
  
  
average\_sales\_per\_product[average\_sales\_per\_product.Category=="Slow"].describe()  
  
  
# ## b. What are your criteria for separating Fast, Medium and Slow Stores? Why?  
  
# In[41]:  
  
  
# Weekly\_sales of each store  
weekly\_sales\_store = non\_promo\_sales.groupby(['StoreCode', "Weekofyear"])['SalesQuantity'].sum().reset\_index()  
  
# Weekly mean sales for each store  
average\_sales\_per\_store = weekly\_sales\_store.groupby('StoreCode')['SalesQuantity'].mean().reset\_index()  
average\_sales\_per\_store.columns = ['StoreCode', 'AvgWeeklySales']  
  
# Seperating fast, medium and slow items  
average\_sales\_per\_store['Rank'] = average\_sales\_per\_store['AvgWeeklySales'].rank(ascending=False)  
average\_sales\_per\_store['Category'] = pd.qcut(average\_sales\_per\_store['Rank'], 3, labels=['Fast', 'Medium', 'Slow'])  
average\_sales\_per\_store  
  
  
# In[42]:  
  
  
average\_sales\_per\_store[average\_sales\_per\_store.Category=="Fast"].describe()  
  
  
# In[43]:  
  
  
average\_sales\_per\_store[average\_sales\_per\_store.Category=="Medium"].describe()  
  
  
# In[44]:  
  
  
average\_sales\_per\_store[average\_sales\_per\_store.Category=="Slow"].describe()  
  
  
# In[45]:  
  
  
# Combining product groups with the data  
assignment4\_1a\_df = assignment4\_1a\_df.merge(average\_sales\_per\_product[['ProductCode', 'Category']], on='ProductCode', how='left')  
assignment4\_1a\_df.rename(columns={'Category': 'ProductCategory'}, inplace=True)  
  
# Combining store groups with the data  
assignment4\_1a\_df = assignment4\_1a\_df.merge(average\_sales\_per\_store[['StoreCode', 'Category']], on='StoreCode', how='left')  
assignment4\_1a\_df.rename(columns={'Category': 'StoreCategory'}, inplace=True)  
  
print(assignment4\_1a\_df.head())  
  
  
# ## c. Which items experienced the biggest sale increase during promotions?  
  
# In[46]:  
  
  
# Sales averages with promotion and non-promotion periods  
promo\_sales = assignment4\_1a\_df[assignment4\_1a\_df['Promotion'] != 'NoPromo']  
non\_promo\_sales = assignment4\_1a\_df[assignment4\_1a\_df['Promotion'] == 'NoPromo']  
  
# Sale averages base on product  
promo\_avg\_sales = promo\_sales.groupby('ProductCode')['SalesQuantity'].mean().reset\_index()  
non\_promo\_avg\_sales = non\_promo\_sales.groupby('ProductCode')['SalesQuantity'].mean().reset\_index()  
  
# Combining to calculate sale increase  
sales\_increase = promo\_avg\_sales.merge(non\_promo\_avg\_sales, on='ProductCode', suffixes=('\_promo', '\_non\_promo'))  
sales\_increase['SalesIncrease'] = sales\_increase['SalesQuantity\_promo'] - sales\_increase['SalesQuantity\_non\_promo']  
  
# Sort by sale increase  
sales\_increase = sales\_increase.sort\_values(by='SalesIncrease', ascending=False)  
print(sales\_increase.head(10))  
  
  
# ## d. Are there stores that have higher promotion reaction?  
  
# In[47]:  
  
  
# Sales averages with promotion and non-promotion periods based on stores  
promo\_avg\_sales\_store = promo\_sales.groupby('StoreCode')['SalesQuantity'].mean().reset\_index()  
non\_promo\_avg\_sales\_store = non\_promo\_sales.groupby('StoreCode')['SalesQuantity'].mean().reset\_index()  
  
# Combining to calculate sale increase   
store\_sales\_increase = promo\_avg\_sales\_store.merge(non\_promo\_avg\_sales\_store, on='StoreCode', suffixes=('\_promo', '\_non\_promo'))  
store\_sales\_increase['SalesIncrease'] = store\_sales\_increase['SalesQuantity\_promo'] - store\_sales\_increase['SalesQuantity\_non\_promo']  
  
# Sort by sale increase  
store\_sales\_increase = store\_sales\_increase.sort\_values(by='SalesIncrease', ascending=False)  
print(store\_sales\_increase.head(10))  
  
  
# ## e. What is the biggest effect explaining sales change during promotions?  
  
# In[48]:  
  
  
assignment4\_1a\_df.info()  
  
  
# In[49]:  
  
  
assignment4\_1a\_df["Promotion"].unique()  
  
  
# assignment4\_1a\_df = assignment4\_1a\_df[assignment4\_1a\_df["Promotion"].isin(['Promo1', 'Promo2', 'Promo3', 'Promo4'])]  
# assignment4\_1a\_df["Promotion"].unique()  
  
# In[50]:  
  
  
assignment4\_1a\_df  
  
  
# In[51]:  
  
  
dummies\_promo = pd.get\_dummies(assignment4\_1a\_df[["Promotion"]])  
dummies\_cat = pd.get\_dummies(assignment4\_1a\_df[["ProductCategory","StoreCategory"]])  
  
  
# ## Model using the first 4 promotions  
  
# In[54]:  
  
  
dummies\_promo = pd.get\_dummies(assignment4\_1a\_df[["Promotion"]])  
dummies\_cat = pd.get\_dummies(assignment4\_1a\_df[["ProductCategory","StoreCategory"]])  
assignment4\_1a\_df\_model = assignment4\_1a\_df.copy()  
assignment4\_1a\_df\_model = pd.concat([assignment4\_1a\_df\_model, dummies\_promo],axis=1)  
assignment4\_1a\_df\_model["Promotion\_Promo5"] = 0  
assignment4\_1a\_df\_model["Promotion\_Promo6"] = 0  
assignment4\_1a\_df\_model = pd.concat([assignment4\_1a\_df\_model, dummies\_cat],axis=1)  
X = assignment4\_1a\_df\_model.drop(["Date","SalesQuantity","Promotion","ProductCategory","StoreCategory"],axis=1)  
y = assignment4\_1a\_df\_model['SalesQuantity']  
  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.25, random\_state=42)  
  
# Hyperparameter optimization  
param\_grid = {  
 'n\_estimators': [100, 200],  
 'max\_depth': [3, 4, 5],  
 'learning\_rate': [0.01, 0.1],  
 'subsample': [0.8, 1.0],  
 'colsample\_bytree': [0.8, 1.0]  
}  
  
# Mode  
xgb = XGBRegressor(tree\_method = "hist")  
  
# GridSearchCV  
grid\_search = GridSearchCV(estimator=xgb, param\_grid=param\_grid,  
 scoring='neg\_mean\_squared\_error', cv=3, verbose=False, n\_jobs=1)  
  
# Training model  
grid\_search.fit(X\_train, y\_train)  
  
# Best parameters  
best\_params = grid\_search.best\_params\_  
print(f"Best parameters: {best\_params}")  
  
# Test  
best\_model = grid\_search.best\_estimator\_  
y\_pred = best\_model.predict(X\_test)  
  
  
# In[55]:  
  
  
assignment4\_1a\_df\_model.info()  
  
  
# In[56]:  
  
  
# Mean squared error  
mse = mean\_squared\_error(y\_test, y\_pred)  
# R-squared (R²)  
r2\_promo4 = r2\_score(y\_test, y\_pred)  
print(f"Mean Squared Error: {mse}")  
print(f"R-squared: {r2\_promo4}")  
  
  
# In[57]:  
  
  
plt.scatter(y\_test, y\_pred)  
  
  
# In[58]:  
  
  
X = sm.add\_constant(X)  
model = sm.OLS(y, X).fit()  
print(model.summary())  
  
  
# In[59]:  
  
  
# Extract and interpret the coefficients to find the largest effect  
coefficients = model.params  
coefficients\_sorted = coefficients.sort\_values(ascending=False)  
print("Largest factors affecting sales change during promotions:")  
print(coefficients\_sorted.head())  
  
  
# In[60]:  
  
  
# Get feature importance  
importance = best\_model.feature\_importances\_  
  
# Plot feature importance  
plot\_importance(best\_model)  
plt.show()  
  
# To print the feature importance in a sorted way  
feature\_importance = pd.DataFrame({  
 'Feature': X.columns,  
 'Importance': importance  
}).sort\_values(by='Importance', ascending=False)  
  
print(feature\_importance)  
  
  
# ## f. Is there any significant difference between promotion impacts of the Fast versus Slow items?  
  
# In[62]:  
  
  
# Fast and slow items based on promotion periods  
product\_categories = assignment4\_1a\_df.groupby(['Promotion', 'ProductCategory'])['SalesQuantity'].mean().reset\_index()  
  
# Bar plot  
plt.figure(figsize=(12, 6))  
sns.barplot(x='Promotion', y='SalesQuantity', hue='ProductCategory', data=product\_categories)  
plt.title('Promotion impacts of the Fast versus Slow items')  
plt.ylabel('Average sales quantity')  
plt.xlabel('Promotion')  
plt.legend(title='Product category')  
plt.show()  
  
  
# In[63]:  
  
  
#A/B Test  
fast\_products = assignment4\_1a\_df[assignment4\_1a\_df['ProductCategory'] == 'Fast']  
slow\_products = assignment4\_1a\_df[assignment4\_1a\_df['ProductCategory'] == 'Slow']  
  
  
fast\_promo\_sales = fast\_products[fast\_products['Promotion'] != 'NoPromo']['SalesQuantity']  
slow\_promo\_sales = slow\_products[slow\_products['Promotion'] != 'NoPromo']['SalesQuantity']  
  
# T-Test  
t\_stat, p\_value = ttest\_ind(fast\_promo\_sales, slow\_promo\_sales)  
  
print(f"T-Stat: {t\_stat}, P-Value: {p\_value}")  
  
  
# ## g. Is there any significant difference between promotion impacts of the Fast versus Slow stores?  
  
# In[78]:  
  
  
# Fast and slow items based on promotion periods  
product\_categories = assignment4\_1a\_df.groupby(['Promotion', 'StoreCategory'])['SalesQuantity'].mean().reset\_index()  
  
# Bar plot  
plt.figure(figsize=(12, 6))  
sns.barplot(x='Promotion', y='SalesQuantity', hue='StoreCategory', data=product\_categories)  
plt.title('Promotion impacts of the Fast versus Slow Stores')  
plt.ylabel('Average sales quantity')  
plt.xlabel('Promotion')  
plt.legend(title='Store category')  
plt.show()  
  
  
# In[64]:  
  
  
# Same approach for stores  
fast\_stores = assignment4\_1a\_df[assignment4\_1a\_df['StoreCategory'] == 'Fast']  
slow\_stores = assignment4\_1a\_df[assignment4\_1a\_df['StoreCategory'] == 'Slow']  
fast\_promo\_sales\_store = fast\_stores[fast\_stores['Promotion'] != 'NoPromo']['SalesQuantity']  
slow\_promo\_sales\_store = slow\_stores[slow\_stores['Promotion'] != 'NoPromo']['SalesQuantity']  
  
t\_stat\_store, p\_value\_store = ttest\_ind(fast\_promo\_sales\_store, slow\_promo\_sales\_store)  
print(f"T-Stat for Fast and Slow stores: {t\_stat\_store}, P-Value: {p\_value\_store}")  
  
  
# ## Model B Promotion5  
  
# In[65]:  
  
  
promotion\_dates\_df  
  
  
# In[66]:  
  
  
assignment4\_1b\_df = pd.read\_csv("assignment4.1b.csv")  
# Convert dates to date format  
assignment4\_1b\_df['Date'] = pd.to\_datetime(assignment4\_1b\_df['Date'], format='%Y/%m/%d', errors='coerce')  
#Based on second dataset's dates adding promotion dates to main dataset  
assignment4\_1b\_df = pd.merge(assignment4\_1b\_df, promo\_expanded, on='Date', how='left')  
  
# Not promotion days named as NoPromo  
assignment4\_1b\_df['Promotion'].fillna('NoPromo', inplace=True)  
  
print(assignment4\_1b\_df.tail())  
  
  
# In[67]:  
  
  
assignment4\_1b\_df.info()  
  
  
# In[68]:  
  
  
assignment4\_1b\_df.Promotion.unique()  
  
  
# In[69]:  
  
  
real\_observed\_sales\_p5 = assignment4\_1b\_df[(assignment4\_1b\_df["Date"]>= "2015-09-01") & ((assignment4\_1b\_df["Date"]<= "2015-09-06")) ]  
real\_observed\_sales\_p5  
  
  
# In[70]:  
  
  
assignment4\_1b\_df["Month"] = assignment4\_1b\_df["Date"].dt.month  
assignment4\_1b\_df["Year"] = assignment4\_1b\_df["Date"].dt.year  
assignment4\_1b\_df["Dayofmonth"] = assignment4\_1b\_df["Date"].dt.day  
assignment4\_1b\_df["Weekofyear"] = assignment4\_1b\_df["Date"].dt.week  
  
# Sales at non-promotion periods  
non\_promo\_salesb = assignment4\_1b\_df[assignment4\_1b\_df['Promotion'] == 'NoPromo']  
  
# Weekly\_sales of each product  
weekly\_salesb = non\_promo\_salesb.groupby(['ProductCode', "Weekofyear"])['SalesQuantity'].sum().reset\_index()  
  
# Weekly mean sales for each product  
average\_sales\_per\_productb = weekly\_salesb.groupby('ProductCode')['SalesQuantity'].mean().reset\_index()  
average\_sales\_per\_productb.columns = ['ProductCode', 'AvgWeeklySales']  
  
# Seperating fast, medium and slow items  
average\_sales\_per\_productb['Rank'] = average\_sales\_per\_productb['AvgWeeklySales'].rank(ascending=False)  
average\_sales\_per\_productb['Category'] = pd.qcut(average\_sales\_per\_productb['Rank'], 3, labels=['Fast', 'Medium', 'Slow'])  
  
# Weekly\_sales of each store  
weekly\_sales\_storeb = non\_promo\_salesb.groupby(['StoreCode', "Weekofyear"])['SalesQuantity'].sum().reset\_index()  
  
# Weekly mean sales for each store  
average\_sales\_per\_storeb = weekly\_sales\_storeb.groupby('StoreCode')['SalesQuantity'].mean().reset\_index()  
average\_sales\_per\_storeb.columns = ['StoreCode', 'AvgWeeklySales']  
  
# Seperating fast, medium and slow items  
average\_sales\_per\_storeb['Rank'] = average\_sales\_per\_storeb['AvgWeeklySales'].rank(ascending=False)  
average\_sales\_per\_storeb['Category'] = pd.qcut(average\_sales\_per\_storeb['Rank'], 3, labels=['Fast', 'Medium', 'Slow'])  
  
  
# Combining product groups with the data  
assignment4\_1b\_df = assignment4\_1b\_df.merge(average\_sales\_per\_productb[['ProductCode', 'Category']], on='ProductCode', how='left')  
assignment4\_1b\_df.rename(columns={'Category': 'ProductCategory'}, inplace=True)  
  
# Combining store groups with the data  
assignment4\_1b\_df = assignment4\_1b\_df.merge(average\_sales\_per\_storeb[['StoreCode', 'Category']], on='StoreCode', how='left')  
assignment4\_1b\_df.rename(columns={'Category': 'StoreCategory'}, inplace=True)  
  
  
  
print(assignment4\_1b\_df.head())  
  
  
# In[71]:  
  
  
assignment4\_1a\_df\_model.columns  
  
  
# In[72]:  
  
  
dummies\_promo = pd.get\_dummies(assignment4\_1b\_df[["Promotion"]])  
dummies\_cat = pd.get\_dummies(assignment4\_1b\_df[["ProductCategory","StoreCategory"]])  
assignment4\_1b\_df\_model = assignment4\_1b\_df.copy()  
assignment4\_1b\_df\_model = pd.concat([assignment4\_1b\_df\_model, dummies\_promo],axis=1)  
assignment4\_1b\_df\_model = assignment4\_1b\_df\_model.assign(Promotion\_Promo1=0, Promotion\_Promo2=0, Promotion\_Promo3=0, Promotion\_Promo4=0)  
assignment4\_1b\_df\_model = assignment4\_1b\_df\_model.rename(columns={"Promotion\_Promo6 ":"Promotion\_Promo6"})  
  
columnsb = list(assignment4\_1b\_df\_model.columns[:-6])+['Promotion\_Promo1', 'Promotion\_Promo2',   
 'Promotion\_Promo3', 'Promotion\_Promo4', 'Promotion\_Promo5', 'Promotion\_Promo6']  
  
assignment4\_1b\_df\_model = assignment4\_1b\_df\_model[columnsb]  
  
  
assignment4\_1b\_df\_model = pd.concat([assignment4\_1b\_df\_model, dummies\_cat],axis=1)  
Xb = assignment4\_1b\_df\_model.drop(["Date","SalesQuantity","Promotion","ProductCategory","StoreCategory"],axis=1)  
yb = assignment4\_1b\_df\_model['SalesQuantity']  
  
# Test  
best\_model = grid\_search.best\_estimator\_  
y\_predb = best\_model.predict(Xb)  
  
  
# In[73]:  
  
  
real\_observed\_sales\_p5  
  
  
# In[74]:  
  
  
predicted\_yb\_df = pd.DataFrame(data = y\_predb, index = assignment4\_1b\_df\_model["Date"])  
predicted\_yb\_df = predicted\_yb\_df[(predicted\_yb\_df.index >= "2015-09-01") & (predicted\_yb\_df.index <= "2015-09-06")]  
predicted\_yb\_df  
  
  
# In[75]:  
  
  
# Mean squared error  
mse5 = mean\_squared\_error(real\_observed\_sales\_p5["SalesQuantity"].values, predicted\_yb\_df.values)  
# R-squared (R²)  
r2\_promo5 = r2\_score(real\_observed\_sales\_p5["SalesQuantity"].values, predicted\_yb\_df.values)  
print(f"Mean Squared Error: {mse5}")  
print(f"R-squared: {r2\_promo5}")  
  
  
# ### What measure would you use for goodness of fit?  
  
# ### How good is your model developed in step 1?  
  
# ### What are the main problem points causing bad fits?  
  
# ### What would you change in step 1?  
  
# ## Bonus: Is there any significant difference in item return rates after promotions?  
  
# In[76]:  
  
  
# Distribution of Sales for each Promotion  
plt.figure(figsize=(10, 6))  
sns.boxplot(x='Promotion', y='SalesQuantity', data=assignment4\_1a\_df, palette='Set3')  
plt.title('Distribution of Sales for each Promotion')  
plt.ylabel('Sales quantity')  
plt.xlabel('Promotion types')  
plt.show()  
  
  
# In[ ]: