Case

Ender Erkaya

July 2024

Introduction

In this case, our aim is to forecast promotion bump in order to give insight to customer about how promotion affects stores and products. For this purpose, we have a dataset includes sales of store item pairs for each day in a period that consists promotion days and regular days. Our aim is to model stores and products using training data in order to estimate their promotion responses.(Part A) Next, we predict the promotion bump for stores and products sales in the test data and validate with actual results and measure the performance of the model we developed in part A.

This report is organized as follows: Section A summarizes all work done in order to cluster products and stores using train data and analyses and statistics. Section B reports our prediction using models derived in Section A and validation results, performances in the test data. Later, report summarizes important findings, results, suggestions and further works in Conclusion.

Section A

Average weekly sale per store is calculated for each products. Below figure demonstrates the result for each product. According to the figure, most of the

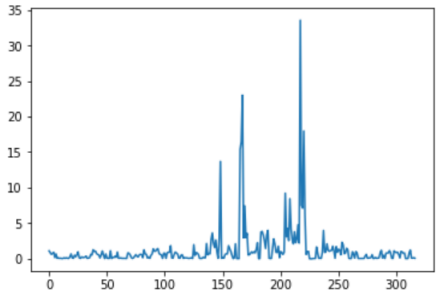


Figure 1: Average Weekly Sales Per Stores

set typically lie in interval $(0, 1)$. But, there are spikes, ie untypical values, outliers that are above 5. This could be regarded as exceptional sets as well as fastest products while modelling. In order to further investigate, histogram is plotted below:

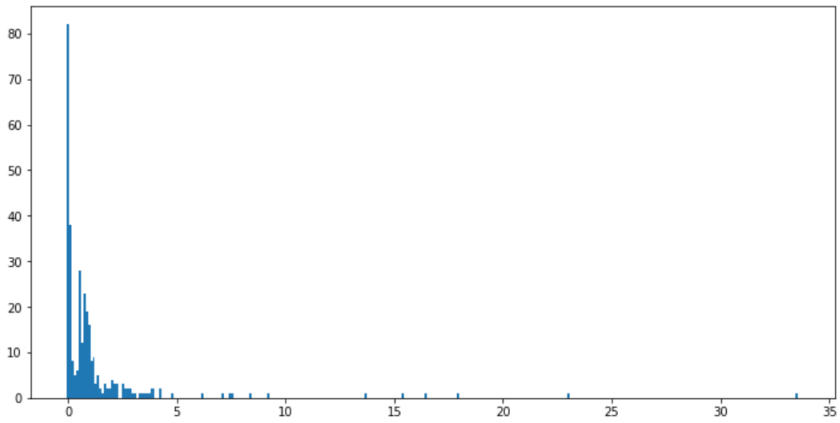


Figure 2: Histogram

A.a

One alternative could be to model the distribution as a combination of three distribution, ie 1 exponential(slow), 1 gaussian(for middle), 1 exponential(fast) or 3 different gaussians. But I chose to model the distribution as an exponential distribution and try to fit roughly in order to determine 3 different clusters. For the purpose, mean, median and standard deviation values are calculated at first. Then, a further estimation is made by getting rid of outliers.

$$\hat{\sigma}_{xrobust} = \hat{\sigma}_x(x[0 < x < x_{median} + 2 * x_{std}])$$

$$\hat{\mu}_{xrobust} = \hat{\mu}_x(x[0 < x < x_{median} + 2 * x_{std}])$$

After estimating parameters, we have $\hat{\mu}_{robust} = 0.80$, $\hat{\sigma}_{robust} = 0.98$ and $median = 0.58$. The values are roughly corresponding an exponential distribution with $\hat{\lambda} = 1.1$. The fitted exponential distribution versus normalized histogram is given in the below figure: To cluster the set, I decided to use 1st quartile and 3rd quartile values of the distribution. Because, 1st quartile represents lowest 0.25 values above 3rd represent highest 0.25 values.

$$x_{slow,threshold} = ln(4/3)/\lambda = 0.26$$

$$x_{fast,threshold} = ln(4)/\lambda = 1.26$$

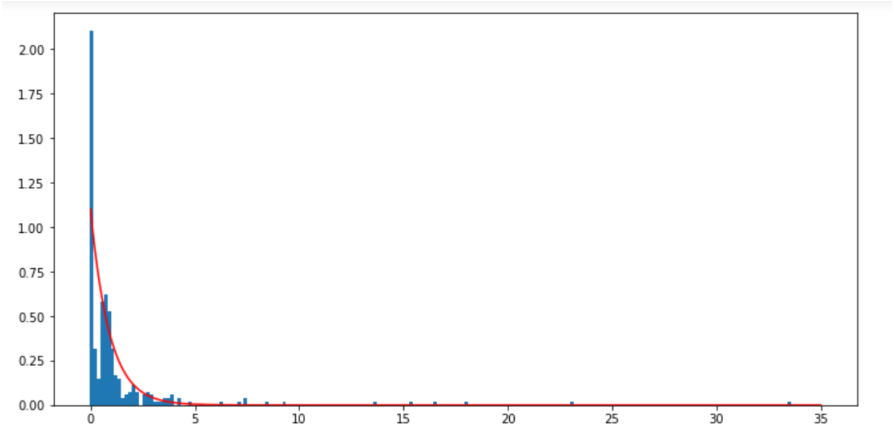


Figure 3: Exponential Distribution Fitting

According to above rules, the decision regions are demonstrated in the below figure:

$$0.26 < x_{medium} < 1.26$$

In the result, there are 128 members in the slow product cluster, 126 elements

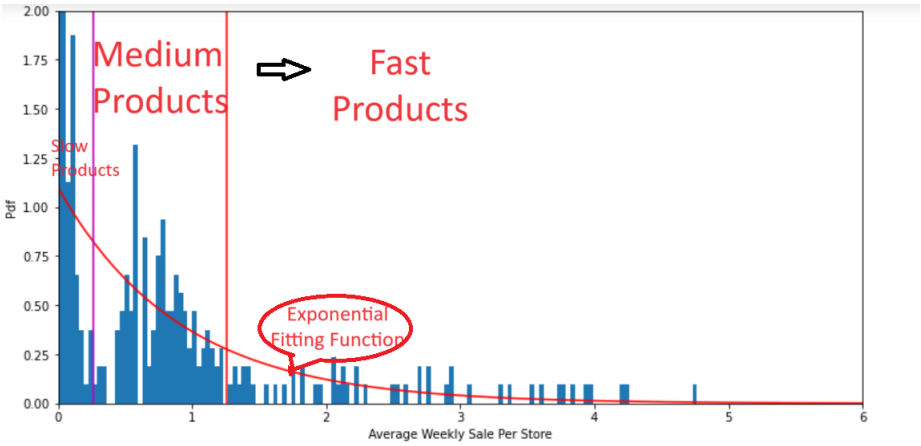


Figure 4: Products Decision Regions

in the medium product cluster and 63 elements in the fast product cluster.

A.b

We approach similar as in the product case. To model the distribution, three different approach is tried. First, a gaussian kernel is fitted by estimat-

ing mean and standard deviation from the observations. Second, a Gaussian mixture approach is fitted using scikit learn library of python and below figure demonstrates the decision boundaries and the fitting. The resulting decision

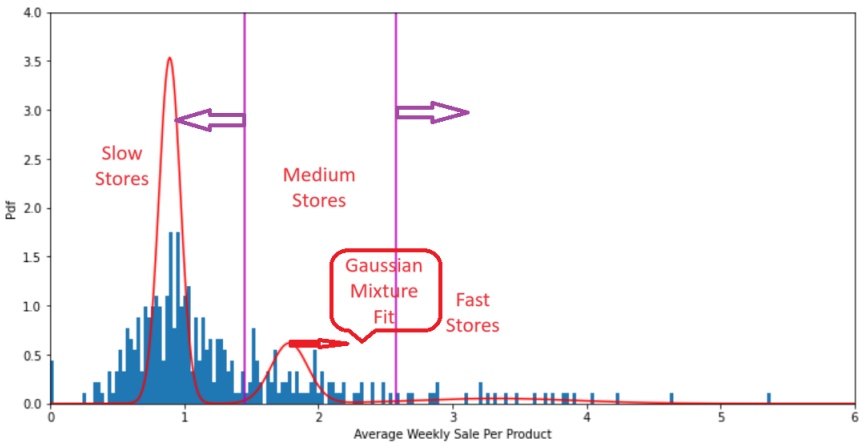


Figure 5: Gaussian Mixture Fitting of Stores

boundaries are 1.45 and 2.58.
As a third approach, a lognormal distribution fitting is executed by estimating parameters from the distribution. Fitting and decision thresholds are demonstrated in the figure below: We observe lognormal distribution fitting represent

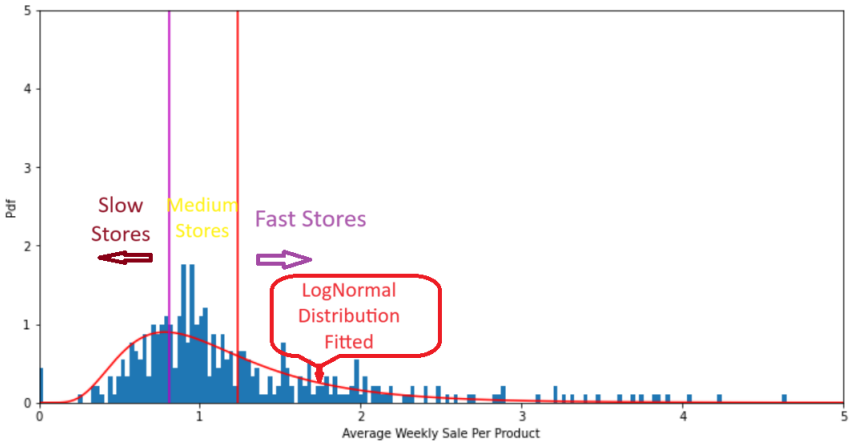


Figure 6: Lognormal Distribution Fitting

better the distribution. It is also more reasonable in case of being one mixture rather than different mixtures. Otherwise, stores have different categorization, it is more logical for stores to be represented by one mixture or distribution.

- Estimated parameters of lognormal distribution is $\mu = 7e-3$ and $\sigma = 0.5$
- In order to divide distribution into 3 clusters, we use quantile function of lognormal distribution.

$$\text{Choose Slow if } x < Quantile(0.33)$$

$$\text{Choose Medium if } Quantile(0.33) < x < Quantile(0.66)$$

$$\text{Choose Fast if } x > Quantile(0.66)$$

- The thresholds are 0.80 and 1.24.

A.c

Increase in quantity by promotion is plotted below: We observe the highest

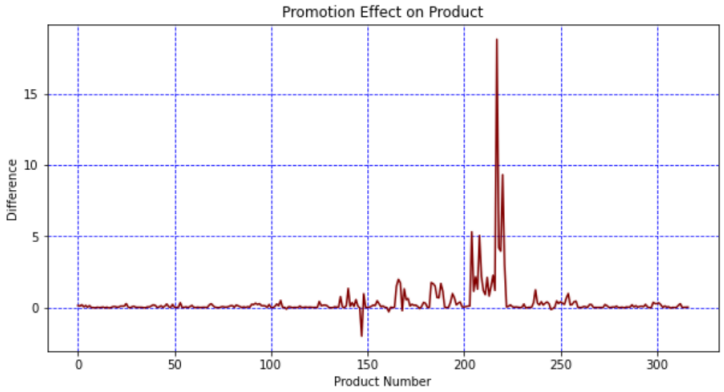


Figure 7: Increase by Promotion

increase in the products are from the fast selling cluster, ie 205-220 region. To further understand the effect, we need to see normalized increase of the difference. Below figure demonstrates the normalized version of the difference, ie normalized by nonpromotion period.

- In absolute increase, the products $\{205, 207, 209, 210, 213, 216, 218, 219, 220, 221, 222\}$ are highest increased products and all of them are already **Fast Items Cluster**, means already in the highest selling group.
- In **ratio**, products $\{13, 193, 229, 231, 268, 291\}$ are highest increased products by promotion, that are increased above 75% after promotion. All products are from the group **Slow Items**
- The products $\{160, 163, 165, 182, 226, 227, 228, 309\}$ are total negative products, ie returns>sales. They **returned positive** from negative sales(returns) after promotion. After promotion, all products have positive sale.

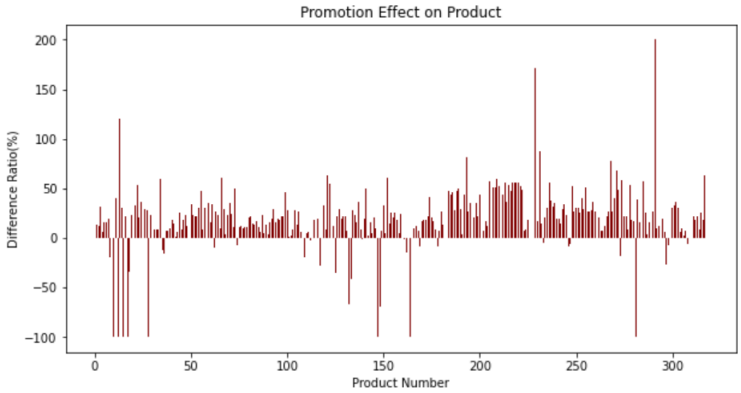


Figure 8: Increase Ratio by Promotion on Products Sale(Only Positive Products

- Although there are negative affected products, the effects are very small compared to the positive effects. The only significant reduction is for product 148, which is reduced by 2 after promotion.
- Products $\{205, 207, 209, 210, 213, 218, 219, 220, 221\}$ are affected positive highest both in magnitude and in ratio($> 50\%$) by promotion.

A.d

The difference by promotion for each store is plotted in the figure. In the plot,

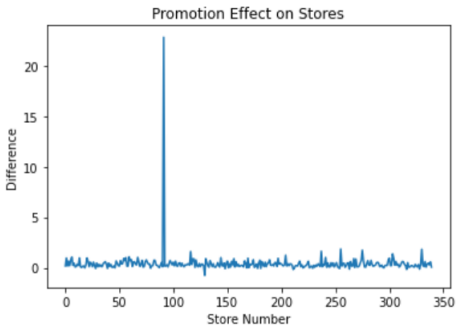


Figure 9: Increase in Stores

there is a significant store that performs outstandingly. Store 92 has highest promotion reaction and can be regarded as an outlier in the stores set. Stores $\{92, 117, 205, 238, 256, 276, 304, 331\}$ have higher reaction than others($> +2\sigma$) to promotion.

A.e

For both Stores and Products, the biggest effect in sales change is due to the cluster type of the Stores and Products. Fast Cluster results more sales change than Slow Cluster. When correlations are calculated, we observe that promotion increase is highly correlated with product sales in nonpromotion periods, ie Fast products increase more.(Correlation Coefficient=0.83). For Stores, there is slightly positive correlations between sales and promotion increase.(Correlation Coefficient= 0.22). This means Fast Stores tend to increase more of their sales compared to Slow Stores. But relationship is weaker than Products Case. Hence, the biggest effect explaining the sale increase is Product Cluster.

A.f

Fast Items are affected in magnitude higher than Slow Items. As Fast Items are increased by 1.5 in average by promotion, Slow Items are increased 0.015 in average by promotion. When we look at in ratio, ie normalized, while Fast Items are increased by 29% in sale, Slow Items are increased by 16%. The results suggest that Fast Items are affected more than Slow Items by promotion. This observation may due that Fast Items are already admirable and popular.

A.g

Fast Stores affected more than slow stores. Below figure summarizes the promotion effect on Fast vs Slow Stores. Even we exclude Store 92, which has

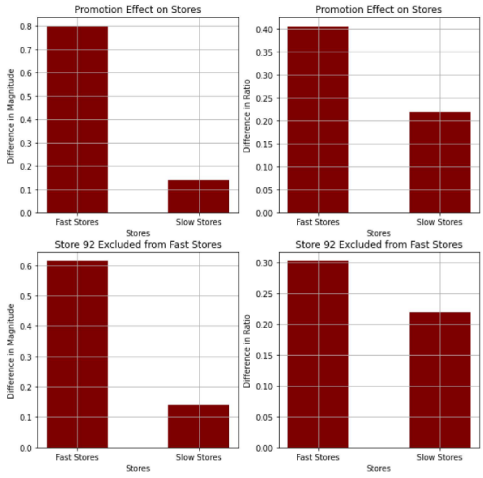


Figure 10: Promotion Effect on Stores

significantly highest change after promotion, Fast Stores react higher than Slow Stores significantly.

Seciton B

B.1 Products Sales Forecasting

In this section, we first predict our products promotion bump and then measure the difference to real results in the test data. For the purpose, we first looked at nonpromoted period in the test data and cluster the products according to the rule we derived in the first section. After clustering, we predict sale increase by ratio and sale increase by magnitude. Belonging to which class, the prediction is appointed as the mean of the sale increase of that class. Below figure demonstrates the predicted increase by ratio versus real increase by ratio for products. Figure 12 demonstrates the predicted increase by magnitude and

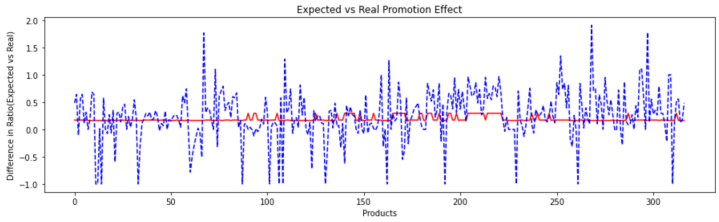


Figure 11: Expected vs Real Promotion Effect(Ratio Increase) on Products

real increase by magnitude. After prediction Root Mean Squared Error is cal-

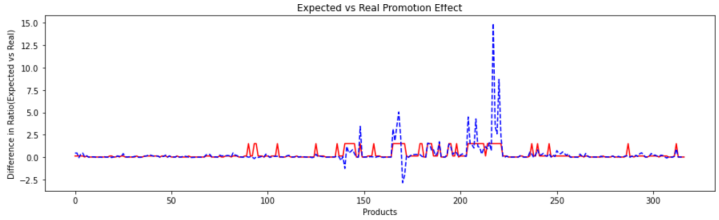


Figure 12: Expected vs Real Promotion Effect(Magnitude Increase) on Products

culated between the predictions vs actual promotion increase calculated from the test data.
In order to evaluate the performance of the model we derived in section A, we need to determine how would it be without the model in A. For the purpose, we can only make one prediction as the mean of the promotion increase without making clustering. Without clustering, we measure RMSE performances of the prediction in order to compare with the performance of strategy A. Below plots show the comparison of RMSE performances of the both strategies.

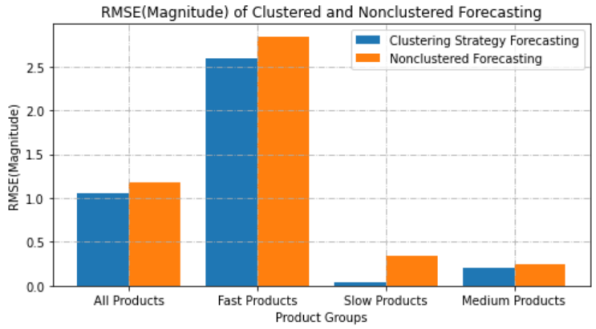


Figure 13: RMSE Comparison of Strategies(Increase by Magnitude on Products)

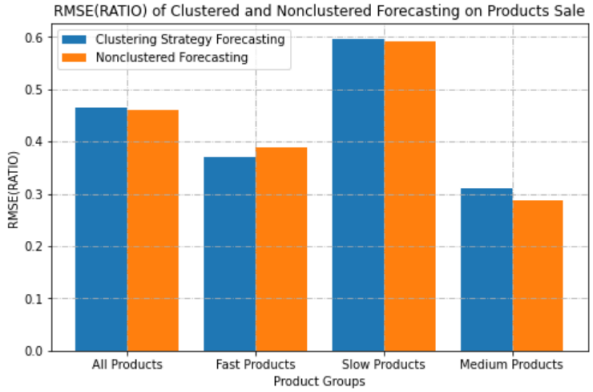


Figure 14: RMSE Comparison of Strategies(Increase by Ratio on Products)

- For each product, average sale per store is calculated and the promotion bump on it is estimated by both in magnitude and in ratio.
- Our magnitude promotion bump prediction using strategy in part A seem to be working efficiently. Because, there is significant decrease in RMSE compared to the strategy without clustering, although it is another simple strategy using the data.
- Our ratio increase promotion bump seems to be not working, inefficient. It may due that ratio is harder to estimate because of its distribution. Only for Fast Product Cluster, ratio estimator seems to be working.
- In test data, there is some items that is not sold during the promotion period. This may due to size of test data. Since, we have only one promotion period in the test data, which is 6 days, it is more vulnerable to disturbances, outliers. Increasing the positive(promotion data) in the test data may yield to measure performance better.

- For magnitude estimation, especially Fast and Small Groups are predicted better than Medium Products.

B.2 Stores Sales Forecasting

Same strategy is applied for stores. RMSE performances are compared between the Strategy A and NonClustered Strategy.

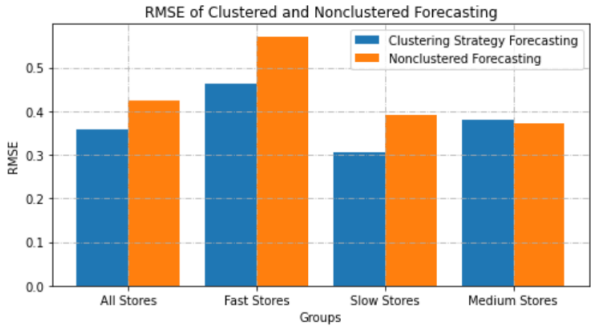


Figure 15: RMSE Comparison of Strategies(Increase by Magnitude on Stores)

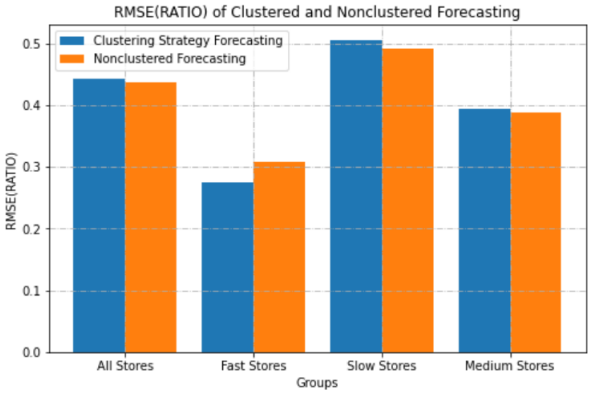


Figure 16: RMSE Comparison of Strategies(Increase by Ratio on Stores)

- As similar to Products Part, our strategy to predict promotion increase by magnitude seems to be working on the contrary to ratio. Thanks to the Strategy in Part A, we are able to reduce RMSE significantly for estimation of magnitude increase by promotion for stores. It means clustering the stores is meaningful for predicting the sale increase by promotion.

- Similar to the previous part, our estimator on the ratio is not working. Only for Fast Stores, we are able to predict the ratio increase better using the clustering. Therefore, we should use ratio predicter only for Fast Stores and not use for other clusters. One approach can be to use ratio estimator if a store belongs to Fast Store Cluster and use Magnitude Estimator otherwise.

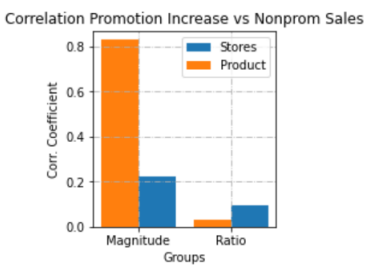
Conclusion

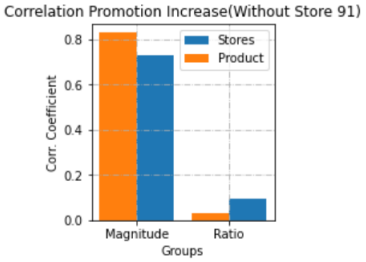
In this case, we tackle the data includes the sales of stores for each day for each product in a period where includes a promotion days and regular days. First, we observe and analyze a train data that includes both promotion and regular days. For analysis, we make clustering of products and stores in three different groups:Fast, Medium, Slow. We theorize that each group tends to react different to promotion. Thanks to clustering model and rules we developed in part A, we make forecasting and validation through the test data. Two different forecastings are tried: Forecasting by ratio increase, forecasting by magnitude increase. We measure their RMSE performances, distance metric to real promotion increase in the test data. We compare RMSE results to the case when we would not make clustering. At the end, we observe that we could decrease RMSE by forecasting by magnitude increase on the contrary to forecasting by ratio increase. We are not able to make forecast by using promotion bump ratio estimation.

- Average sales per store is calculated for each product and average sales per product is calculated for each store. First random variable(product) is modeled as an exponential distribution and their quantiles are used for clustering. Second random variable(stores) is modeled as a lognormal distribution and its quantiles are used to decide clusters.

- In test data, for forecasting, we first look at nonpromotion periods and classify, cluster stores and products according to the rules we derived in part A. Then we predict the promotion sale increase of stores and products and compare them with actual results in the test data. At the end, we observe that our strategy in part A seem to work fine and efficiently decrease RMSE results by forecasting the magnitude increase. On the contrary, forecasting the ratio increase did not work. To scrath the reason why ratio forecasting did not work, we need to observe below figure demonstrating correlation results. Although, mean ratio increase for Fast Stores and Products are higher than Slow Products and Stores, there is not correlation between ratio increase and average regular sales. Hence, we should use to forecast magnitude increase instead of ratio increase.

- While calculating ratio promotion increase, items that have total negative sale in regular periods, ie more return prohibits the calculation. Hence,





(a) Correlation Coefficient W Store 91 (b) Correlation Coefficient W.out Store 91

we needed to calculate for positive items. This also makes harder to use ratio data.

- We observe that there is a strong correlation between product sales and their promotion increases. Fast products are highly probable to increase more of their sales in promotion periods. For stores, we observe a slightly positive correlation between sales and promotion increase, if we include store 91, which has exceptional performance. When we regard store 91 as an outlier and ignore it, we see that there is strong correlation between store promotion bump and their regular sales.

- Below figure demonstrates the plot comparing predicted promotion bump vs real promotion bump for stores.

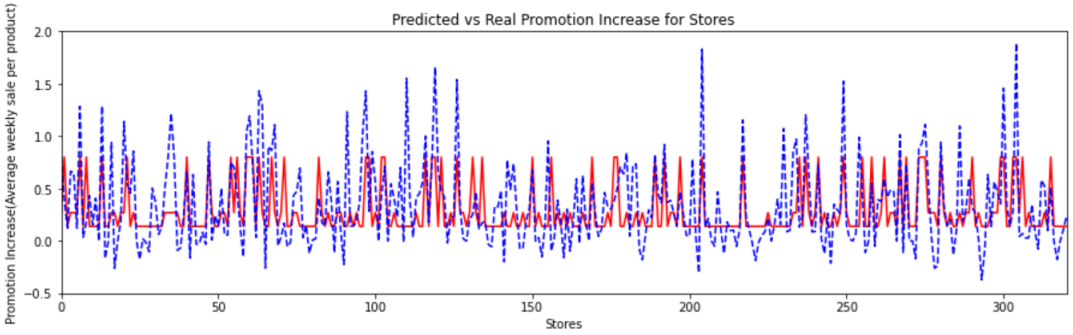


Figure 18: Promotion Bump Forecasting for Stores

- We compare RMSE performance of Strategy A with a basic strategy, when we do not make clustering and predict with a mean average calculated by train data. By comparison, we see that we can reduce RMSE by making clustering $0.07, 15.5\%$ for stores and $0.13, 10.8\%$ for products.(Figure 19)

- The products $160, 163, 165, 182, 226, 227, 228, 309$ are total negative products, ie returns>sales. They returned positive from negative sales(returns)

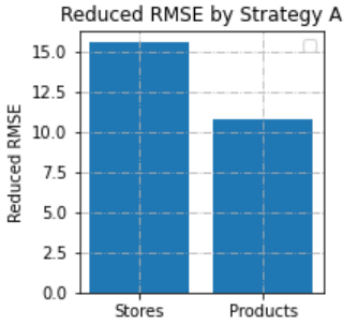


Figure 19: Reduced RMSE by Model A

after promotion. After promotion, all products have positive sale.

- The products affected more by promotion, both in magnitude and in ratio are $205, 207, 209, 210, 213, 218, 219, 220, 221$. All of them are from same category **Product Group1:A, Product Group 2:5**. It means **(A,5) Category** tends to affect specially by promotion. In promotion periods, they would need more supply. Products $\{184, 185, 186, 187, 188, 189, 190\}$ are affected more than other products by promotion. They are also from the same **Category (A,5)**.

- After **Category (A,5)**, there is another category performs better than other products: **Category (G,4)**, which includes products $\{166, 167, 168, 169, 170, 171, 172\}$

- Two categories respond more than others: **(A,5)** and **(G,4)**. Both of them are Fast Products and successfully classified as Fast Products by algorithm developed in part A.

- Since test data has very few days of promotion periods, some products have experienced no sale in promotion days, reduced from positive. It may means test data is short to observe and validate the effect of our model. It would be better to increase promotion days, ie promotion/regular days ratio in the test data in order to represent products behavior more accurately.

- For further work, average sale per week is calculated for each product-store pairs, which results a matrix. To estimate its promotion bump would be harder. One possible approach is to use our clustering model in part A, resulting total of 9 clusters such as (Fast Product, Slow Store),(Medium Product, Fast Store) etc. We may try an estimator like:

$$\hat{\Delta}_{prom} = Sale_{regular} \times \sqrt{\hat{r}_{product}\hat{r}_{store}}$$

$$\hat{\Delta}_{prom} = \frac{\hat{\Delta}_{store} + \hat{\Delta}_{product}}{2}$$

where we multiply magnitude increase by geometric mean of ratio predictions of each separate cluster derived in part A.
Another alternative is to make clustering in 2D for Product-Store domain and predict using this clustering method. For this purpose, other clustering methods such as dbscan, kmeans can be used.

1 Appendix

1.1 Jupyter Codes

Listing 1: Project code

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
#                              Case
## Ender Erkaya
### 7/15/2024
#Import
import numpy as np
import pandas as pd
import math
from datetime import datetime as dt
import matplotlib.pyplot as plt
import scipy as sp
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
# PART A
...
#Load Data
df_sales      = pd.read_csv('assignment4.1a.csv')
df_promotions = pd.read_csv('PromotionDates.csv')
...
#Visualize Dataset
print(df_sales.head())
print(df_promotions)
print(df_sales.info())
print(df_promotions.info())
(num_rows_sales,num_columns_sales)=df_sales.shape
print("There are", num_rows_sales, "observations in the
    dataset")
print(df_sales.describe())
print(True in df_sales.duplicated())
...
df_sales["ProductCode"].nunique()
...
print(df_sales)
max(list(df_sales[df_sales["ProductCode"] == 217].
    SalesQuantity))
...
num_stores    = df_sales["StoreCode"].nunique()
num_products = df_sales["ProductCode"].nunique()
print("There are", num_stores, "stores")
print("There are", num_products, "products")
...
```

```
def change_datetime(x):
    return dt.strptime(x, '%Y-%m-%d')
def change_datetime2(x):
    return dt.strptime(x, '%m/%d/%Y')
def add_zeros(x):
    return "0"+x
...
df_sales["Date"] = df_sales["Date"].map(change_datetime)
...
df_promotions.loc[:4,"StartDate"]=df_promotions.loc[:4,"
    StartDate"].map(add_zeros)
df_promotions.loc[:4,"EndDate"]=df_promotions.loc[:4,"
    EndDate"].map(add_zeros)
print(df_promotions)
...
import copy
df_promotions4 = copy.deepcopy(df_promotions.loc[:3,:])
df_promotions4["StartDate"]=df_promotions4["StartDate"].
    map(change_datetime2)
df_promotions4["EndDate"]=df_promotions4["EndDate"].map(
    change_datetime2)
...
num_promotions = 4
promotion_index=[]
for i in range(num_promotions):
    temp = df_sales[(df_sales["Date"]>= df_promotions4.
        loc[i,"StartDate"]) & (df_sales["Date"] <=
        df_promotions4.loc[i,"EndDate"])]
    promotion_index.extend(temp.index)
...
def weird_division(n, d):
    return n / d if d else 0
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
## Product Clustering
average_weekly_prom_sale_per_store     = np.zeros(
    num_products)
average_weekly_nonprom_sale_per_store = np.zeros(
    num_products)
average_weekly_sale_per_store          = np.zeros(
    num_products)
for i in range(num_products):
    prod_code = i+1
    if prod_code%50==0:
        print(prod_code)
    df_prod = df_sales[df_sales["ProductCode"]==prod_code
        ]
```

```
    df_prod_prom     = copy.deepcopy(df_prod[df_prod.index
        .isin(promotion_index)])
    df_prod_nonprom = copy.deepcopy(df_prod[~df_prod.
        index.isin(promotion_index)])
    num_prom_days    = df_prod_prom["Date"].nunique() ##33
    num_nonprom_days= df_prod_nonprom["Date"].nunique() #
        #176
    total_prom_sale = df_prod_prom["SalesQuantity"].sum()
    total_nonprom_sale = df_prod_nonprom["SalesQuantity"
        ].sum()
    total_sale = df_prod["SalesQuantity"].sum()
    average_weekly_prom_sale_per_store[prod_code-1] = (
        weird_division(total_prom_sale,num_prom_days)*7)/
        num_stores
    average_weekly_nonprom_sale_per_store[prod_code-1] =
        (weird_division(total_nonprom_sale,
        num_nonprom_days)*7)/num_stores
    average_weekly_sale_per_store[prod_code-1] = (
        weird_division(total_sale,(num_prom_days+
        num_nonprom_days))*7)/num_stores
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
### Visualize Average Sales of Products in NonPromotion
    Period

print(average_weekly_nonprom_sale_per_store.argmax())
plt.plot(average_weekly_nonprom_sale_per_store)
plt.figure(figsize=(12,6),edgecolor='blue')
plt.hist(average_weekly_nonprom_sale_per_store,100,
    density=1)
plt.show()
...
mean_nonprom_prod    = np.mean(
    average_weekly_nonprom_sale_per_store[
    average_weekly_nonprom_sale_per_store>0])
median_nonprom_prod = np.median(
    average_weekly_nonprom_sale_per_store[
    average_weekly_nonprom_sale_per_store>0])
std_nonprom_prod     = np.std(
    average_weekly_nonprom_sale_per_store[
    average_weekly_nonprom_sale_per_store>0])
print("Mean:", mean_nonprom_prod)
print("Median:", median_nonprom_prod)
print("Standard Deviation:", std_nonprom_prod)
...
robust_std_nonprom_prod     = np.std(
    average_weekly_nonprom_sale_per_store[(0<
```

```
    average_weekly_nonprom_sale_per_store)&(
    average_weekly_nonprom_sale_per_store<
    median_nonprom_prod+2*std_nonprom_prod)])
print(robust_std_nonprom_prod)
robust_mean_nonprom_prod = np.mean(
    average_weekly_nonprom_sale_per_store[(0<
    average_weekly_nonprom_sale_per_store)&(
    average_weekly_nonprom_sale_per_store<
    median_nonprom_prod+2*std_nonprom_prod)])
print(robust_mean_nonprom_prod)
...
...
slow_threshold      = 0.26
medium_lower_bound = slow_threshold
medium_upper_bound = 1.26
fast_threshold      = medium_upper_bound
...
slow_product_index    = np.where(
    average_weekly_nonprom_sale_per_store<=slow_threshold)
medium_product_index = np.where((
    average_weekly_nonprom_sale_per_store>
    medium_lower_bound)&(
    average_weekly_nonprom_sale_per_store<
    medium_upper_bound))
fast_product_index    = np.where(
    average_weekly_nonprom_sale_per_store>=fast_threshold)
...
slow_product_positive_index = np.where((0<
    average_weekly_nonprom_sale_per_store)&(
    average_weekly_nonprom_sale_per_store<=slow_threshold)
    )
...
print(np.array(slow_product_index).shape)
print(np.array(slow_product_positive_index).shape)
...
print(np.shape(slow_product_index))
print(np.shape(medium_product_index))
print(np.shape(fast_product_index))
print(np.shape(average_weekly_nonprom_sale_per_store))
...
#def exponential_distribution(x):
#    return lambda*np.exp(-lambda * x)
lambada = 1.1
xx = np.linspace(0,10,1000)
plt.figure(figsize=(10,5),edgecolor='blue')
plt.plot(xx, lambada*np.exp(-lambada * xx),'r')
```

```
plt.hist(average_weekly_nonprom_sale_per_store,1000,
    density=True)
plt.axvline(x = 0.26, color = 'm', label = 'axvline - full height')
plt.axvline(x = 1.26, color = 'r', label = 'axvline - full height')
plt.xlabel("Average Weekly Sale Per Store")
plt.ylabel("Pdf")
ax = plt.gca()
ax.set_xlim([0, 6])
ax.set_ylim([0, 2])
...
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
## Stores Clustering

average_weekly_prom_sale_per_product      = np.zeros(
    num_stores)
average_weekly_nonprom_sale_per_product = np.zeros(
    num_stores)
average_weekly_sale_per_product           = np.zeros(
    num_stores)
for i in range(num_stores):
    store_code = i+1
    if store_code%50==0:
        print(store_code)
    df_store = df_sales[df_sales["StoreCode"]==store_code
        ]
    df_store_prom     = copy.deepcopy(df_store[df_store.
        index.isin(promotion_index)])
    df_store_nonprom = copy.deepcopy(df_store[~df_store.
        index.isin(promotion_index)])
    num_prom_days    = df_store_prom["Date"].nunique() #
        #33
    num_nonprom_days= df_store_nonprom["Date"].nunique()
        ##176
    total_prom_sale = df_store_prom["SalesQuantity"].sum
        ()
    total_nonprom_sale = df_store_nonprom["SalesQuantity"
        ].sum()
    total_sale = df_store["SalesQuantity"].sum()
    average_weekly_prom_sale_per_product[store_code-1] =
        (weird_division(total_prom_sale,num_prom_days)*7)/
        num_products
  ...  average_weekly_nonprom_sale_per_product[store_code
    -1] = (weird_division(total_nonprom_sale,
    num_nonprom_days)*7)/num_products
```

```
    average_weekly_sale_per_product[store_code-1] = (
        weird_division(total_sale,(num_prom_days+
        num_nonprom_days))*7)/num_products
...
### Visualize Average Sales of Store in Non Promotion
    Period
...
print(np.where(average_weekly_nonprom_sale_per_product<0)
    )
...
print(average_weekly_nonprom_sale_per_product.argmax())
plt.plot(average_weekly_nonprom_sale_per_product)
plt.figure(figsize=(12,6),edgecolor='blue')
plt.hist(average_weekly_nonprom_sale_per_product,100,
    density=1)
plt.grid(color='b', linestyle='--')
plt.xlabel("Average Weekly Sale Per Product")
plt.ylabel("pdf")
plt.show()
...
...
mean_nonprom_store   = np.mean(
    average_weekly_nonprom_sale_per_product)
median_nonprom_store = np.median(
    average_weekly_nonprom_sale_per_product)
std_nonprom_store    = np.std(
    average_weekly_nonprom_sale_per_product)
print("Mean:", mean_nonprom_store)
print("Median:", median_nonprom_store)
print("Standard Deviation:", std_nonprom_store)
...
## Fitting Gaussian Distribution
def gaussian(x, mu, sig):
    return (1.0 / (np.sqrt(2.0 * np.pi) * sig) * np.exp(-
        np.power((x - mu) / sig, 2.0) / 2))
...
mu    = median_nonprom_store
sigma = std_nonprom_store
...
xx = np.linspace(0,10,1000)
plt.figure(figsize=(12,6),edgecolor='blue')
plt.plot(xx, gaussian(xx,mu,sigma),'r')
plt.hist(average_weekly_nonprom_sale_per_product,50,
    density=True)
plt.xlabel("Average Weekly Sale Per Product")
plt.ylabel("Pdf")
```

```
ax = plt.gca()
ax.set_xlim([0, 10])
ax.set_ylim([0, 3])
...
lower_threshold = mean_nonprom_store - std_nonprom_store
upper_threshold = mean_nonprom_store + std_nonprom_store
print(lower_threshold)
print(upper_threshold)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
### Different Approach:Gaussian Mixture

from sklearn.mixture import GaussianMixture
gm = GaussianMixture(n_components=3, random_state=0).fit(
    average_weekly_nonprom_sale_per_product.reshape(-1, 1)
    )
[mu1,mu2,mu3] = gm.means_
[sigma1,sigma2,sigma3] = gm.covariances_.reshape(-1,1)
print(mu1,sigma1)
print(mu2,sigma2)
print(mu3,sigma3)

gm.weights_

xx = np.linspace(0,10,1000)
clusters = gm.predict(xx.reshape(-1,1))+1
plt.plot(xx,clusters)
temp_index  = np.array(np.where(np.gradient(clusters))).
    reshape(-1,)+1
lower_threshold = xx[temp_index[0]]
upper_threshold = xx[temp_index[-1]]
print(lower_threshold)
print(upper_threshold)

xx = np.linspace(0,10,1000)

plt.figure(figsize=(12,6),edgecolor='blue')
plt.hist(average_weekly_nonprom_sale_per_product,300,
    density=True)
plt.plot(xx, gm.weights_[0]*gaussian(xx,mu1,sigma1)+gm.
    weights_[1]*gaussian(xx,mu2,sigma2)+gm.weights_[2]*
    gaussian(xx,mu3,sigma3),'r')
plt.axvline(x = lower_threshold, color = 'm', label = '
    axvline - full height')
plt.axvline(x = upper_threshold, color = 'm', label = '
    axvline - full height')
plt.grid()
```

```
plt.xlabel("Average Weekly Sale Per Product")
plt.ylabel("Pdf")
ax = plt.gca()
ax.set_xlim([0, 6])
ax.set_ylim([0, 4])

### Different Approach: Log Normal Distribution

mu_lognormal = 7e-3 #estimated from median
sigma_lognormal = math.sqrt(2*(math.log(
    mean_nonprom_store)-mu_lognormal))
mode_nonprom_store = np.array(sp.stats.mode(
    average_weekly_nonprom_sale_per_product))
print(sigma_lognormal)
...
def lognormal(x, mu, sig):
    return (1.0 / (x*(np.sqrt(2.0 * np.pi) * sig)) * np.
        exp(-np.power((np.log(x) - mu) / sig, 2.0) / 2))

xx = np.linspace(1e-16,10,1000)
...
plt.figure(figsize=(12,6),edgecolor='blue')
plt.hist(average_weekly_nonprom_sale_per_product,200,
    density=True)
plt.plot(xx, lognormal(xx,mu_lognormal,sigma_lognormal),'
    r')
plt.grid()
plt.xlabel("Average Weekly Sale Per Product")
plt.ylabel("Pdf")
ax = plt.gca()
ax.set_xlim([0, 6])
ax.set_ylim([0, 2])
...
mu_lognormal = 7e-3
sigma_lognormal = 0.5
...
xx = np.linspace(1e-16,10,1000)
...
plt.figure(figsize=(12,6),edgecolor='blue')
plt.hist(average_weekly_nonprom_sale_per_product,200,
    density=True)
plt.plot(xx, lognormal(xx,mu_lognormal,sigma_lognormal),'
    r')
plt.xlabel("Average Weekly Sale Per Product")
plt.ylabel("Pdf")
plt.grid()
```

```
ax = plt.gca()
ax.set_xlim([0, 6])
ax.set_ylim([0, 2])
...
from scipy.special import erfinv, erf
lower_store_threshold = math.exp(mu_lognormal+math.sqrt
    (0.5)*erfinv(2*0.33-1))
print(lower_store_threshold)
upper_store_threshold = math.exp(mu_lognormal+math.sqrt
    (0.5)*erfinv(2*0.66-1))
print(upper_store_threshold)
...
xx = np.linspace(1e-16,10,1000)
...
plt.figure(figsize=(12,6),edgecolor='blue')
plt.hist(average_weekly_nonprom_sale_per_product,200,
    density=True)
plt.plot(xx, lognormal(xx,mu_lognormal,sigma_lognormal),'
    r')
plt.axvline(x = lower_threshold, color = 'm', label = '
    axvline - full height')
plt.axvline(x = upper_threshold, color = 'r', label = '
    axvline - full height')
plt.grid()
plt.xlabel("Average Weekly Sale Per Product")
plt.ylabel("Pdf")
ax = plt.gca()
ax.set_xlim([0, 5])
ax.set_ylim([0, 2])
...
slow_store_index    = np.where(
    average_weekly_nonprom_sale_per_product<=
    lower_store_threshold)
medium_store_index = np.where((
    average_weekly_nonprom_sale_per_product>
    lower_store_threshold)&(
    average_weekly_nonprom_sale_per_product<
    upper_store_threshold))
fast_store_index    = np.where(
    average_weekly_nonprom_sale_per_product>=
    upper_store_threshold)
...
print(np.array(slow_store_index).shape)
print(np.array(medium_store_index).shape)
print(np.array(fast_store_index).shape)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
## c Promotion Effects on Items
...
difference_average_sale_product = (
    average_weekly_prom_sale_per_store -
    average_weekly_nonprom_sale_per_store)
difference_average_sale_ratio_product = ((
    difference_average_sale_product-1e-16)/(abs(
    average_weekly_nonprom_sale_per_store)+1e-16))
...
print(np.mean(difference_average_sale_product))
...
difference_average_sale_product[162]
print(average_weekly_prom_sale_per_store[162])
print(average_weekly_nonprom_sale_per_store[162])
print(average_weekly_sale_per_store[162])
...
# creating the dataset
fig = plt.figure(figsize = (10, 5))
...
# creating the bar plot
plt.plot(range(num_products),100*
    difference_average_sale_product,color ='maroon')
plt.grid(color='b', linestyle='--')
plt.xlabel("Product Number")
plt.ylabel("Difference")
plt.title("Promotion Effect on Product")
plt.show()
...
negative_affected_products_index = np.array(np.where(
    difference_average_sale_product<0))
print(negative_affected_products_index)
print(difference_average_sale_product[147])
print(average_weekly_prom_sale_per_store[147])
print(average_weekly_nonprom_sale_per_store[147])
...
highest_increased_products = np.array(np.where(
    difference_average_sale_product>2))+1
print(highest_increased_products)
print(np.array(fast_product_index)+1)
...
positive_product_index = np.array(np.where(0<
    average_weekly_nonprom_sale_per_store))
print(positive_product_index.shape)
...
print(difference_average_sale_ratio_product[
    positive_product_index].shape)
```

```
...
# creating the dataset
fig = plt.figure(figsize = (10, 5))
...
# creating the bar plot
plt.bar(positive_product_index.reshape(-1,)+1,100*
    difference_average_sale_ratio_product[
    positive_product_index].reshape(-1,),color ='maroon',
        width = 0.6)
...
plt.xlabel("Product Number")
plt.ylabel("Difference Ratio(%)")
plt.title("Promotion Effect on Product")
plt.show()
...
difference_average_sale_ratio_product[12]
...
highest_increased_ratio_products_index = np.where(
    difference_average_sale_ratio_product>50)
print(np.array(highest_increased_ratio_products_index)+1)
print(np.array(fast_product_index)+1)
...
### There are returned items during nonprom periods that
    should be further investigated.
...
returned_products_nonprom_index = np.array(np.where(
    average_weekly_nonprom_sale_per_store<0))
print(returned_products_nonprom_index+1)
### Check if their sales increased
temp = average_weekly_prom_sale_per_store -
    average_weekly_nonprom_sale_per_store
print(temp[returned_products_nonprom_index])
...
print(np.array(np.where(
    average_weekly_prom_sale_per_store<0)))
...
print(np.where((difference_average_sale_ratio_product>40)
    &(difference_average_sale_product>2)))
...
### The sales of returned products (160, 163, 165, 182,
    226, 227, 228, 309) changes to positive by promotion.
    Also there are items that returned negative during
    promotion period.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

## d Stores Reaction by Promotion
```

```
...
difference_average_sale_stores = (
    average_weekly_prom_sale_per_product -
    average_weekly_nonprom_sale_per_product)
difference_average_sale_ratio_stores = ((
    difference_average_sale_stores-1e-16)/(
    average_weekly_nonprom_sale_per_product+1e-16))
...
plt.plot(difference_average_sale_stores)
plt.xlabel("Store Number")
plt.ylabel("Difference")
plt.title("Promotion Effect on Stores")
plt.grid()
print(difference_average_sale_stores[91])
print(average_weekly_prom_sale_per_product[91])
print(average_weekly_nonprom_sale_per_product[91])
print(np.mean(difference_average_sale_stores))
print(np.std(difference_average_sale_stores[
    difference_average_sale_stores<20]))
...
highest_affected_stores = np.array(np.where(
    difference_average_sale_stores>1.1))+1
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

## f. Fast vs Slow Products Change by Promotion
...
fast_products_nonprom_sale =
    average_weekly_nonprom_sale_per_store[
    fast_product_index]
slow_products_nonprom_sale =
    average_weekly_nonprom_sale_per_store[
    slow_product_index]
slow_products_positive_nonprom_sale =
    average_weekly_nonprom_sale_per_store[
    slow_product_positive_index]
medium_products_nonprom_sale =
    average_weekly_nonprom_sale_per_store[
    medium_product_index]
fast_products_prom_sale =
    average_weekly_prom_sale_per_store[fast_product_index]
slow_products_prom_sale =
    average_weekly_prom_sale_per_store[slow_product_index]
slow_products_positive_prom_sale =
    average_weekly_prom_sale_per_store[
    slow_product_positive_index]
medium_products_prom_sale =
```

```
    average_weekly_prom_sale_per_store[
    medium_product_index]
difference_fast_products_sale = fast_products_prom_sale -
     fast_products_nonprom_sale
difference_slow_products_sale = slow_products_prom_sale -
     slow_products_nonprom_sale
difference_slow_products_positive_sale =
    slow_products_positive_prom_sale -
    slow_products_positive_nonprom_sale
difference_medium_products_sale =
    medium_products_prom_sale -
    medium_products_nonprom_sale
...
print(np.mean(difference_fast_products_sale))
print(np.mean(difference_slow_products_sale))
print(np.mean(difference_slow_products_positive_sale))
print(np.mean(difference_medium_products_sale))
...
difference_fast_products_sale_ratio =
    difference_fast_products_sale / (
    fast_products_nonprom_sale+1e-16)
difference_slow_products_sale_ratio = (
    difference_slow_products_positive_sale) / (
    slow_products_positive_nonprom_sale)
difference_medium_products_sale_ratio =
    difference_medium_products_sale / (
    medium_products_nonprom_sale)
print(np.mean(difference_fast_products_sale_ratio))
print(np.mean(difference_slow_products_sale_ratio))
print(np.mean(difference_medium_products_sale_ratio))
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

## g. Fast vs Slow Stores Change by Promotion
...
fast_stores_nonprom_sale =
    average_weekly_nonprom_sale_per_product[
    fast_store_index]
slow_stores_nonprom_sale =
    average_weekly_nonprom_sale_per_product[
    slow_store_index]
medium_stores_nonprom_sale =
    average_weekly_nonprom_sale_per_product[
    medium_store_index]
fast_stores_prom_sale =
    average_weekly_prom_sale_per_product[fast_store_index]
slow_stores_prom_sale =
```

```
    average_weekly_prom_sale_per_product[slow_store_index]
medium_stores_prom_sale =
    average_weekly_prom_sale_per_product[
    medium_store_index]
difference_fast_stores_sale = fast_stores_prom_sale -
    fast_stores_nonprom_sale
difference_slow_stores_sale = slow_stores_prom_sale -
    slow_stores_nonprom_sale
difference_medium_stores_sale = medium_stores_prom_sale -
     medium_stores_nonprom_sale
...
print(np.mean(difference_fast_stores_sale))
print(np.mean(difference_slow_stores_sale))
print(np.mean(difference_medium_stores_sale))
...
difference_fast_stores_sale_ratio =
    difference_fast_stores_sale / (
    fast_stores_nonprom_sale)
difference_slow_stores_sale_ratio = (
    difference_slow_stores_sale-1e-16) / (abs(
    slow_stores_nonprom_sale)+1e-16)
difference_medium_stores_sale_ratio =
    difference_medium_stores_sale / (
    medium_stores_nonprom_sale)
print(np.median(difference_fast_stores_sale_ratio))
print(np.median(difference_slow_stores_sale_ratio))
print(np.median(difference_medium_stores_sale_ratio))
...
print(difference_fast_stores_sale_ratio.shape)
...
temp  = average_weekly_prom_sale_per_product/(abs(
    average_weekly_nonprom_sale_per_product)+1e-16)
temp2 = np.log(temp, out=np.zeros_like(temp, dtype=np.
    float64), where=(temp!=0))
mean_temp = np.mean(temp2)
std_temp = np.std(temp2)
print(mean_temp)
print(std_temp)
lognormal_mean_stores = math.exp(mean_temp+(std_temp**2)
    /2)-1
print(lognormal_mean_stores)
...
temp  = fast_stores_prom_sale/fast_stores_nonprom_sale
temp2 = np.log(temp, out=np.zeros_like(temp, dtype=np.
    float64), where=(temp!=0))
mean_temp_fast = np.mean(temp2)
```

```
std_temp_fast = np.std(temp2)
print(mean_temp_fast)
print(std_temp_fast)
lognormal_mean_fast_stores = math.exp(mean_temp_fast+(
    std_temp_fast**2)/2)-1
print(lognormal_mean_fast_stores)
...
temp  = slow_stores_prom_sale/(slow_stores_nonprom_sale+1
    e-16)
temp2 = np.log(temp, out=np.zeros_like(temp, dtype=np.
    float64), where=(temp!=0))
mean_temp_slow = np.mean(temp2)
std_temp_slow = np.std(temp2)
print(mean_temp_slow)
print(std_temp_slow)
lognormal_mean_slow_stores = math.exp(mean_temp_slow+(
    std_temp_slow**2)/2)-1
print(lognormal_mean_slow_stores)
...
temp  = medium_stores_prom_sale/(
    medium_stores_nonprom_sale+1e-16)
temp2 = np.log(temp, out=np.zeros_like(temp, dtype=np.
    float64), where=(temp!=0))
mean_temp_medium = np.mean(temp2)
std_temp_medium = np.std(temp2)
print(mean_temp_medium)
print(std_temp_medium)
lognormal_mean_medium_stores = math.exp(mean_temp_medium
    +(std_temp_medium**2)/2)-1
print(lognormal_mean_medium_stores)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

### To observe the effect of the Store 92:
...
print(np.array(fast_store_index))
fast_store_index2 = np.delete(fast_store_index ,np.array(
    np.where(np.array(fast_store_index).reshape(-1,) ==
    91)))
print(fast_store_index2)
...
fast_stores_nonprom_sale =
    average_weekly_nonprom_sale_per_product[
    fast_store_index]
fast_stores_nonprom_sale2 =
    average_weekly_nonprom_sale_per_product[
    fast_store_index2]
```

```
slow_stores_nonprom_sale =
    average_weekly_nonprom_sale_per_product[
    slow_store_index]
fast_stores_prom_sale =
    average_weekly_prom_sale_per_product[fast_store_index]
fast_stores_prom_sale2 =
    average_weekly_prom_sale_per_product[fast_store_index2
    ]
slow_stores_prom_sale =
    average_weekly_prom_sale_per_product[slow_store_index]
difference_fast_stores_sale = fast_stores_prom_sale -
    fast_stores_nonprom_sale
difference_fast_stores_sale2 = fast_stores_prom_sale2 -
    fast_stores_nonprom_sale2
difference_slow_stores_sale = slow_stores_prom_sale -
    slow_stores_nonprom_sale
...
print(np.mean(difference_fast_stores_sale2))
print(np.mean(difference_slow_stores_sale))
...
difference_fast_stores_sale_ratio2 =
    difference_fast_stores_sale2 / (
    fast_stores_nonprom_sale2)
difference_slow_stores_sale_ratio = (
    difference_slow_stores_sale[slow_stores_nonprom_sale
    >0]) / (slow_stores_nonprom_sale[
    slow_stores_nonprom_sale>0])
print(np.mean(difference_fast_stores_sale_ratio2))
print(np.mean(difference_slow_stores_sale_ratio))
...
# creating the dataset
fig = plt.figure(figsize = (10, 10))
...
# creating the bar plot
plt.subplot(221)
plt.bar(["Fast Stores","Slow Stores"],[np.mean(
    difference_fast_stores_sale),np.mean(
    difference_slow_stores_sale)],color ='maroon',
        width = 0.5)
plt.grid()
plt.ylabel("Difference in Magnitude")
plt.xlabel("Stores")
plt.title("Promotion Effect on Stores")
...
plt.subplot(222)
plt.bar(["Fast Stores","Slow Stores"],[np.mean(
```

```
    difference_fast_stores_sale_ratio),np.mean(
    difference_slow_stores_sale_ratio)],color ='maroon',
        width = 0.5)
plt.grid()
plt.ylabel("Difference in Ratio")
plt.xlabel("Stores")
plt.title("Promotion Effect on Stores")

plt.subplot(223)
plt.bar(["Fast Stores","Slow Stores"],[np.mean(
    difference_fast_stores_sale2),np.mean(
    difference_slow_stores_sale)],color ='maroon',
        width = 0.5)
plt.grid()
plt.ylabel("Difference in Magnitude")
plt.xlabel("Stores")
plt.title("Store 92 Excluded from Fast Stores")
...
plt.subplot(224)
plt.bar(["Fast Stores","Slow Stores"],[np.mean(
    difference_fast_stores_sale_ratio2),np.mean(
    difference_slow_stores_sale_ratio)],color ='maroon',
        width = 0.5)
plt.grid()
plt.ylabel("Difference in Ratio")
plt.xlabel("Stores")
plt.title("Store 92 Excluded from Fast Stores")
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

# PART B
...
df_future_sales = pd.read_csv('assignment4.1b.csv')
...
#Visualize Dataset
print(df_future_sales.head())
print(df_future_sales.info())
(num_rows_future_sales,num_columns_future_sales)=df_sales
    .shape
print("There are", num_rows_future_sales, "observations
    in the dataset")
print(df_future_sales.describe())
print(True in df_future_sales.duplicated())
...
df_promotions.loc[4,"StartDate"] = '09/01/2015'
df_promotions.loc[4,"EndDate"] = '09/06/2015'
...
```

```
df_promotion5 = copy.deepcopy(df_promotions.iloc[4,:])
df_promotion5["StartDate"]= dt.strptime(df_promotion5["
    StartDate"], '%m/%d/%Y')
df_promotion5["EndDate"]   = dt.strptime(df_promotion5["
    EndDate"], '%m/%d/%Y')
df_promotion5
...
df_future_sales["Date"] = df_future_sales["Date"].map(
    change_datetime)
...
temp = df_future_sales[(df_future_sales["Date"]>=
    df_promotion5["StartDate"]) & (df_future_sales["Date"]
     <= df_promotion5["EndDate"])]
promotion5_index = temp.index
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

## B.1. Product Sales Forecasting
...
average_weekly_nonprom_future_sale_per_store = np.zeros(
    num_products)
...
for i in range(num_products):
     prod_code = i+1
     if prod_code%50==0:
          print(prod_code)
     df_future_prod = df_future_sales[df_future_sales["
         ProductCode"]==prod_code]
     df_prod_future_nonprom = copy.deepcopy(df_future_prod
         [~df_future_prod.index.isin(promotion5_index)])
     num_nonprom_days= df_prod_future_nonprom["Date"].
         nunique()
     total_nonprom_future_sale = df_prod_future_nonprom["
         SalesQuantity"].sum()
     average_weekly_nonprom_future_sale_per_store[
         prod_code-1] = (weird_division(
         total_nonprom_future_sale,num_nonprom_days)*7)/
         num_stores
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

### Predict Product Cluster in the Nonprom Period
...
future_product_cluster    = np.zeros(num_products)
predict_product_ratio     = np.zeros(num_products)
predict_product_increase = np.zeros(num_products)
...
fast_future_product_index = []
```

```
slow_future_product_index = []
medium_future_product_index = []
...
## Predict Products Cluster
for i in range(num_products):
    if (average_weekly_nonprom_future_sale_per_store[i]
       <= 0.26):
        slow_future_product_index.append(i)
        future_product_cluster[i]    = 0
        predict_product_ratio[i]     = 0.262
        predict_product_increase[i] = 0.013
    elif (average_weekly_nonprom_future_sale_per_store[i]
        >= 1.26):
        fast_future_product_index.append(i)
        future_product_cluster[i]    = 2
        predict_product_ratio[i]     = 0.295
        predict_product_increase[i] = 1.51
    else:
        medium_future_product_index.append(i)
        future_product_cluster[i]    = 1
        predict_product_ratio[i]     = 0.171
        predict_product_increase[i] = 0.13
...
product_cluster = np.zeros(num_products)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

## Predict Products Cluster
for i in range(num_products):
    if (average_weekly_nonprom_sale_per_store[i] <= 0.26)
        :
        product_cluster[i] = 0

    elif (average_weekly_nonprom_sale_per_store[i] >=
       1.26):
        product_cluster[i] = 2
    else:
        product_cluster[i] = 1
...
#Predicted Increase Ratio by Products
fig = plt.figure(figsize = (10, 5))
plt.plot(range(num_products),predict_product_increase)
plt.ylabel("Expected Increase In Ratio(%)")
plt.xlabel("Products")
plt.title("Expected Increase Ratio In Products")
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
### Calculate Real Promotion Response of Products
...
average_weekly_prom_future_sale_per_store = np.zeros(
    num_products)
...
for i in range(num_products):
    prod_code = i+1
    if prod_code%50==0:
        print(prod_code)
    df_future_prod = df_future_sales[df_future_sales["
        ProductCode"]==prod_code]
    df_prod_future_prom = copy.deepcopy(df_future_prod[
        df_future_prod.index.isin(promotion5_index)])
    num_prom_days= df_prod_future_prom["Date"].nunique()
    total_prom_future_sale = df_prod_future_prom["
        SalesQuantity"].sum()
    average_weekly_prom_future_sale_per_store[prod_code
        -1] = (weird_division(total_prom_future_sale,
        num_prom_days)*7)/num_stores
...
difference_average_future_sale_product = (
    average_weekly_prom_future_sale_per_store -
    average_weekly_nonprom_future_sale_per_store)
difference_average_future_sale_ratio_product = np.divide(
    difference_average_future_sale_product, abs(
    average_weekly_nonprom_future_sale_per_store), out=np.
    zeros_like(difference_average_future_sale_product,
    dtype=float), where=abs(
    average_weekly_nonprom_future_sale_per_store)!=0)
...
fig = plt.figure(figsize = (10, 5))

# creating the bar plot
plt.bar(range(num_products),100*
    difference_average_future_sale_ratio_product,color ='
    maroon',
        width = 0.6)
...
plt.xlabel("Product-Number")
plt.ylabel("Difference-Ratio(%)")
plt.title("Promotion-Effect-on-Product")
plt.show()
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

# Compare Expected and Actual Increase in the Test Set
fig = plt.figure(figsize = (15, 4))
```

```
# creating the plot
plt.plot(range(num_products), predict_product_increase,'r
    --')
plt.plot(range(num_products),
    difference_average_future_sale_product,'b---')
plt.ylabel("Difference in Ratio(Expected vs Real)")
plt.xlabel("Products")
plt.title("Expected vs Real Promotion Effect")
...
difference_average_future_sale_product[165:171]
...
RMSE_RATIO_PRODUCTS          = math.sqrt(np.mean(np.square(
    predict_product_ratio-
    difference_average_future_sale_ratio_product)))
RMSE_RATIO_FAST_PRODUCTS     = math.sqrt(np.mean(np.square(
    predict_product_ratio[fast_future_product_index]-
    difference_average_future_sale_ratio_product[
    fast_future_product_index])))
RMSE_RATIO_SLOW_PRODUCTS     = math.sqrt(np.mean(np.square(
    predict_product_ratio[slow_future_product_index]-
    difference_average_future_sale_ratio_product[
    slow_future_product_index])))
RMSE_RATIO_MEDIUM_PRODUCTS = math.sqrt(np.mean(np.square(
    predict_product_ratio[medium_future_product_index]-
    difference_average_future_sale_ratio_product[
    medium_future_product_index])))
print(RMSE_RATIO_PRODUCTS)
print(RMSE_RATIO_FAST_PRODUCTS)
print(RMSE_RATIO_SLOW_PRODUCTS)
print(RMSE_RATIO_MEDIUM_PRODUCTS)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

### RMSE PERFORMANCE OF MAGNITUDE PREDICTION OF CLUSTERED
    PRODUCTS
RMSE_PRODUCTS_MAGNITUDE          = math.sqrt(np.mean(np.
    square(predict_product_increase-
    difference_average_future_sale_product)))
RMSE_FAST_PRODUCTS_MAGNITUDE     = math.sqrt(np.mean(np.
    square(predict_product_increase[
    fast_future_product_index]-
    difference_average_future_sale_product[
    fast_future_product_index])))
RMSE_SLOW_PRODUCTS_MAGNITUDE     = math.sqrt(np.mean(np.
    square(predict_product_increase[
    slow_future_product_index]-
```

```
    difference_average_future_sale_product[
    slow_future_product_index])))
RMSE_MEDIUM_PRODUCTS_MAGNITUDE = math.sqrt(np.mean(np.
    square(predict_product_increase[
    medium_future_product_index]-
    difference_average_future_sale_product[
    medium_future_product_index])))
print(RMSE_PRODUCTS_MAGNITUDE)
print(RMSE_FAST_PRODUCTS_MAGNITUDE)
print(RMSE_SLOW_PRODUCTS_MAGNITUDE)
print(RMSE_MEDIUM_PRODUCTS_MAGNITUDE)
...
predict_product_increase[fast_future_product_index]
...
np.mean(difference_average_future_sale_product[
    fast_product_index])
...
### To measure the effect of the model we developed in
    the first section, lets look at what MSE would be if
    we did not make any clustering. If we did not cluster,
     we would decide our sale increase by 0.232 by
    promotion.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

### RMSE PERFORMANCE OF RATIO PREDICTION OF NONCLUSTERED
    PRODUCTS
prediction_ratio_not_clustered      = 0.232
RMSE_PRODUCTS_RATIO_NOT_CLUSTERED          = math.sqrt(np.
    mean(np.square(prediction_ratio_not_clustered-
    difference_average_future_sale_ratio_product)))
RMSE_FAST_PRODUCTS_RATIO_NOT_CLUSTERED     = math.sqrt(np.
    mean(np.square(prediction_ratio_not_clustered-
    difference_average_future_sale_ratio_product[
    fast_future_product_index])))
RMSE_SLOW_PRODUCTS_RATIO_NOT_CLUSTERED     = math.sqrt(np.
    mean(np.square(prediction_ratio_not_clustered-
    difference_average_future_sale_ratio_product[
    slow_future_product_index])))
RMSE_MEDIUM_PRODUCTS_RATIO_NOT_CLUSTERED = math.sqrt(np.
    mean(np.square(prediction_ratio_not_clustered-
    difference_average_future_sale_ratio_product[
    medium_future_product_index])))
print(RMSE_PRODUCTS_RATIO_NOT_CLUSTERED)
print(RMSE_FAST_PRODUCTS_RATIO_NOT_CLUSTERED)
print(RMSE_SLOW_PRODUCTS_RATIO_NOT_CLUSTERED)
print(RMSE_MEDIUM_PRODUCTS_RATIO_NOT_CLUSTERED)
```

```
%%%
```

```
...
plt.bar(X_axis - 0.2, Clustering, 0.4, label = '
    Clustering Strategy Forecasting')
plt.bar(X_axis + 0.2, NonClustered, 0.4, label = '
    Nonclustered Forecasting')
plt.grid(linestyle = "-.")
plt.xticks(X_axis, X)
plt.xlabel("Product Groups")
plt.ylabel("RMSE(Magnitude)")
plt.title("RMSE(Magnitude) of Clustered and Nonclustered
    Forecasting")
plt.legend()
plt.show()
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

### COMPARISON OF RMSE(RATIO) PERFORMANCES OF CLUSTERING
    VS NONCLUSTERED
fig = plt.figure(figsize = (8, 5))
...
X = ['All Products','Fast Products','Slow Products','
    Medium Products']
Clustering      = [RMSE_RATIO_PRODUCTS,
    RMSE_RATIO_FAST_PRODUCTS,RMSE_RATIO_SLOW_PRODUCTS,
    RMSE_RATIO_MEDIUM_PRODUCTS]
NonClustered  = [RMSE_PRODUCTS_RATIO_NOT_CLUSTERED,
    RMSE_FAST_PRODUCTS_RATIO_NOT_CLUSTERED,
    RMSE_SLOW_PRODUCTS_RATIO_NOT_CLUSTERED,
    RMSE_MEDIUM_PRODUCTS_RATIO_NOT_CLUSTERED]
...
X_axis = np.arange(len(X))
...
plt.bar(X_axis - 0.2, Clustering, 0.4, label = '
    Clustering Strategy Forecasting')
plt.bar(X_axis + 0.2, NonClustered, 0.4, label = '
    Nonclustered Forecasting')
plt.grid(linestyle = "-.")
plt.xticks(X_axis, X)
plt.xlabel("Product Groups")
plt.ylabel("RMSE(RATIO)")
plt.title("RMSE(RATIO) of Clustered and Nonclustered
    Forecasting on Products Sale")
plt.legend()
plt.show()
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

## B.2 Stores Sale Forecasting
```

```
...
average_weekly_nonprom_future_sale_per_product = np.zeros
    (num_stores)
...
for i in range(num_stores):
    store_code = i+1
    if store_code%50==0:
        print(store_code)
    df_future_store = df_future_sales[df_future_sales["
        StoreCode"]==store_code]
    df_store_future_nonprom = copy.deepcopy(
        df_future_store[~df_future_store.index.isin(
        promotion5_index)])
    num_nonprom_days= df_store_future_nonprom["Date"].
        nunique()
    total_nonprom_future_sale = df_store_future_nonprom["
        SalesQuantity"].sum()
    average_weekly_nonprom_future_sale_per_product[
        store_code-1] = (weird_division(
        total_nonprom_future_sale,num_nonprom_days)*7)/
        num_products
...
### Cluster the Stores and Predict Promotion Bump
...
future_store_cluster   = np.zeros(num_stores)
predict_store_ratio    = np.zeros(num_stores)
predict_store_increase = np.zeros(num_stores)
...
fast_future_store_index = []
slow_future_store_index = []
medium_future_store_index = []
...
## Predict Store Cluster
## Predict Store Promotion Bump
for i in range(num_stores):
    if (average_weekly_nonprom_future_sale_per_product[i]
        <= 0.8):
        slow_future_store_index.append(i)
        future_store_cluster[i]   = 0
        predict_store_ratio[i]    = 0.17
        predict_store_increase[i] = 0.14
    elif (average_weekly_nonprom_future_sale_per_product[
        i] >= 1.24):
        fast_future_store_index.append(i)
        future_store_cluster[i]   = 2
        predict_store_ratio[i]    = 0.41
```

```
            predict_store_increase[i] = 0.8
        else:
            medium_future_store_index.append(i)
            future_store_cluster[i]    = 1
            predict_store_ratio[i]     = 0.27
            predict_store_increase[i] = 0.27
...
#Predicted Increase Ratio by Stores
fig = plt.figure(figsize = (10, 5))
plt.plot(range(num_stores),predict_store_increase)
plt.ylabel("Expected Increase In Ratio(%)")
plt.xlabel("Stores")
plt.title("Expected Increase Ratio In Stores")
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

### Calculate Real Promotion Response of Stores
...
average_weekly_prom_future_sale_per_product = np.zeros(
    num_stores)
...
for i in range(num_stores):
    store_code = i+1
    if store_code%50==0:
        print(store_code)
    df_future_store = df_future_sales[df_future_sales["
        StoreCode"]==store_code]
    df_store_future_prom = copy.deepcopy(df_future_store[
        df_future_store.index.isin(promotion5_index)])
    num_prom_days= df_store_future_prom["Date"].nunique()
    total_prom_future_sale = df_store_future_prom["
        SalesQuantity"].sum()
    average_weekly_prom_future_sale_per_product[
        store_code-1] = (weird_division(
        total_prom_future_sale,num_prom_days)*7)/
        num_products
...
difference_average_future_sale_store = (
    average_weekly_prom_future_sale_per_product -
    average_weekly_nonprom_future_sale_per_product)
difference_average_future_sale_ratio_store = np.divide(
    difference_average_future_sale_store, abs(
    average_weekly_nonprom_future_sale_per_product), out=
    np.zeros_like(difference_average_future_sale_store,
    dtype=float), where=abs(
    average_weekly_nonprom_future_sale_per_product)!=0)
...
```

```
fig = plt.figure(figsize = (10, 5))
...
# creating the bar plot
plt.bar(range(num_stores),
    difference_average_future_sale_store,color ='maroon',
        width = 0.6)
...
plt.xlabel("Store Number")
plt.ylabel("Difference Ratio(%)")
plt.title("Promotion Effect on Stores")
plt.show()
...
print(average_weekly_prom_future_sale_per_product[338])
print(average_weekly_nonprom_future_sale_per_product
    [338])
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
# Compare Expected and Actual Increase of Stores Sale in
    the Test Set
fig = plt.figure(figsize = (15, 4))
...
# creating the plot
plt.plot(range(num_stores), predict_store_increase, 'r-')
plt.plot(range(num_stores),
    difference_average_future_sale_store, 'b--')
plt.ylabel("Promotion Increase(Average weekly sale per
    product)")
plt.xlabel("Stores")
plt.title("Predicted vs Real Promotion Increase for
    Stores")
ax = plt.gca()
ax.set_xlim([0, 320])
ax.set_ylim([-0.5, 2])
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
### RMSE PERFORMANCE OF RATIO PREDICTION
RMSE_STORES_RATIO          = math.sqrt(np.mean(np.square(
    predict_store_ratio-
    difference_average_future_sale_ratio_store)))
RMSE_FAST_STORES_RATIO     = math.sqrt(np.mean(np.square(
    predict_store_ratio[fast_future_store_index]-
    difference_average_future_sale_ratio_store[
    fast_future_store_index])))
RMSE_SLOW_STORES_RATIO     = math.sqrt(np.mean(np.square(
    predict_store_ratio[slow_future_store_index]-
    difference_average_future_sale_ratio_store[
    slow_future_store_index])))
RMSE_MEDIUM_STORES_RATIO = math.sqrt(np.mean(np.square(
```

```
    predict_store_ratio[medium_future_store_index]-
    difference_average_future_sale_ratio_store[
    medium_future_store_index])))
print(RMSE_STORES_RATIO)
print(RMSE_FAST_STORES_RATIO)
print(RMSE_SLOW_STORES_RATIO)
print(RMSE_MEDIUM_STORES_RATIO)
...
print(predict_store_ratio[medium_future_store_index])
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

### RMSE PERFORMANCE OF MAGNITUDE PREDICTION
RMSE_STORES_MAGNITUDE          = math.sqrt(np.mean(np.
    square(predict_store_increase-
    difference_average_future_sale_store)))
RMSE_FAST_STORES_MAGNITUDE     = math.sqrt(np.mean(np.
    square(predict_store_increase[fast_future_store_index
    ]-difference_average_future_sale_store[
    fast_future_store_index])))
RMSE_SLOW_STORES_MAGNITUDE     = math.sqrt(np.mean(np.
    square(predict_store_increase[slow_future_store_index
    ]-difference_average_future_sale_store[
    slow_future_store_index])))
RMSE_MEDIUM_STORES_MAGNITUDE = math.sqrt(np.mean(np.
    square(predict_store_increase[
    medium_future_store_index]-
    difference_average_future_sale_store[
    medium_future_store_index])))
print(RMSE_STORES_MAGNITUDE)
print(RMSE_FAST_STORES_MAGNITUDE)
print(RMSE_SLOW_STORES_MAGNITUDE)
print(RMSE_MEDIUM_STORES_MAGNITUDE)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

### What if we did not make clustering?
...
prediction_ratio_store_not_clustered      = np.mean(
    difference_average_sale_ratio_stores)
prediction_magnitude_store_not_clustered = np.mean(
    difference_average_sale_stores)

RMSE_STORES_RATIO_NOT_CLUSTERED          = math.sqrt(np.
    mean(np.square(prediction_ratio_store_not_clustered-
    difference_average_future_sale_ratio_store)))
RMSE_FAST_STORES_RATIO_NOT_CLUSTERED     = math.sqrt(np.
    mean(np.square(prediction_ratio_store_not_clustered-
```

```
    difference_average_future_sale_ratio_store[
    fast_future_store_index])))
RMSE_SLOW_STORES_RATIO_NOT_CLUSTERED    = math.sqrt(np.
    mean(np.square(prediction_ratio_store_not_clustered -
    difference_average_future_sale_ratio_store[
    slow_future_store_index])))
RMSE_MEDIUM_STORES_RATIO_NOT_CLUSTERED = math.sqrt(np.
    mean(np.square(prediction_ratio_store_not_clustered -
    difference_average_future_sale_ratio_store[
    medium_future_store_index])))
print(RMSE_STORES_RATIO_NOT_CLUSTERED)
print(RMSE_FAST_STORES_RATIO_NOT_CLUSTERED)
print(RMSE_SLOW_STORES_RATIO_NOT_CLUSTERED)
print(RMSE_MEDIUM_STORES_RATIO_NOT_CLUSTERED)
...
RMSE_STORES_MAGNITUDE_NOT_CLUSTERED          = math.sqrt(np
    .mean(np.square(
    prediction_magnitude_store_not_clustered -
    difference_average_future_sale_store)))
RMSE_FAST_STORES_MAGNITUDE_NOT_CLUSTERED    = math.sqrt(np
    .mean(np.square(
    prediction_magnitude_store_not_clustered -
    difference_average_future_sale_store[
    fast_future_store_index])))
RMSE_SLOW_STORES_MAGNITUDE_NOT_CLUSTERED    = math.sqrt(np
    .mean(np.square(
    prediction_magnitude_store_not_clustered -
    difference_average_future_sale_store[
    slow_future_store_index])))
RMSE_MEDIUM_STORES_MAGNITUDE_NOT_CLUSTERED = math.sqrt(np
    .mean(np.square(
    prediction_magnitude_store_not_clustered -
    difference_average_future_sale_store[
    medium_future_store_index])))
print(RMSE_STORES_MAGNITUDE_NOT_CLUSTERED)
print(RMSE_FAST_STORES_MAGNITUDE_NOT_CLUSTERED)
print(RMSE_SLOW_STORES_MAGNITUDE_NOT_CLUSTERED)
print(RMSE_MEDIUM_STORES_MAGNITUDE_NOT_CLUSTERED)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

### COMPARISON OF RMSE(MAGNITUDE) PERFORMANCES OF
    CLUSTERING VS NONCLUSTERED
fig = plt.figure(figsize = (8, 4))
...
X = ['All-Stores','Fast-Stores','Slow-Stores','Medium-
    Stores']
```

```
Clustering     = [RMSE_STORES_MAGNITUDE,
    RMSE_FAST_STORES_MAGNITUDE,RMSE_SLOW_STORES_MAGNITUDE,
    RMSE_MEDIUM_STORES_MAGNITUDE]
NonClustered   = [RMSE_STORES_MAGNITUDE_NOT_CLUSTERED,
    RMSE_FAST_STORES_MAGNITUDE_NOT_CLUSTERED,
    RMSE_SLOW_STORES_MAGNITUDE_NOT_CLUSTERED,
    RMSE_MEDIUM_STORES_MAGNITUDE_NOT_CLUSTERED]
...
X_axis = np.arange(len(X))
...
plt.bar(X_axis - 0.2, Clustering, 0.4, label = '
    Clustering Strategy Forecasting')
plt.bar(X_axis + 0.2, NonClustered, 0.4, label = '
    Nonclustered Forecasting')
plt.grid(linestyle = "--")
plt.xticks(X_axis, X)
plt.xlabel("Groups")
plt.ylabel("RMSE")
plt.title("RMSE of Clustered and Nonclustered Forecasting
    ")
plt.legend()
plt.show()
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

### COMPARISON OF RMSE(RATIO) PERFORMANCES OF CLUSTERING
    VS NONCLUSTERED
fig = plt.figure(figsize = (8, 5))

X = ['All Stores','Fast Stores','Slow Stores','Medium
    Stores']
Clustering     = [RMSE_STORES_RATIO,RMSE_FAST_STORES_RATIO
    ,RMSE_SLOW_STORES_RATIO,RMSE_MEDIUM_STORES_RATIO]
NonClustered   = [RMSE_STORES_RATIO_NOT_CLUSTERED,
    RMSE_FAST_STORES_RATIO_NOT_CLUSTERED,
    RMSE_SLOW_STORES_RATIO_NOT_CLUSTERED,
    RMSE_MEDIUM_STORES_RATIO_NOT_CLUSTERED]
...
X_axis = np.arange(len(X))
...
plt.bar(X_axis - 0.2, Clustering, 0.4, label = '
    Clustering Strategy Forecasting')
plt.bar(X_axis + 0.2, NonClustered, 0.4, label = '
    Nonclustered Forecasting')
plt.grid(linestyle = "--")
plt.xticks(X_axis, X)
plt.xlabel("Groups")
```

```
plt.ylabel("RMSE(RATIO)")
plt.title("RMSE(RATIO) of Clustered and Nonclustered
    Forecasting")
plt.legend()
plt.show()
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

# CONCLUSION
...
stores_corr_prom    = np.corrcoef(
    difference_average_sale_stores,
    average_weekly_nonprom_sale_per_product)[0,1]
products_corr_prom = np.corrcoef(
    difference_average_sale_product,
    average_weekly_nonprom_sale_per_store)[0,1]
print(stores_corr_prom)
print(products_corr_prom)
...
temp1 = np.delete(difference_average_sale_stores,91)
temp2 = np.delete(average_weekly_nonprom_sale_per_product
    ,91)
stores_corr_prom_without91 = np.corrcoef(temp1, temp2)
    [0,1]
print(stores_corr_prom_without91)
...
stores_corr_prom_ratio    = (np.corrcoef(
    difference_average_sale_ratio_stores,
    average_weekly_nonprom_sale_per_product))[0,1]
                              ...
products_corr_prom_ratio = np.corrcoef(
    difference_average_sale_ratio_product,
    average_weekly_nonprom_sale_per_store)[0,1]
print(stores_corr_prom_ratio)
print(products_corr_prom_ratio)
...
fig = plt.figure(figsize = (6, 3))

plt.subplot(1,2,1)
X = ['Magnitude','Ratio']
Product = [products_corr_prom, products_corr_prom_ratio]
Stores = [stores_corr_prom, stores_corr_prom_ratio]
X_axis = np.arange(len(X))
plt.bar(X_axis + 0.2, Stores, 0.4, label = 'Stores')
plt.bar(X_axis - 0.2, Product, 0.4, label = 'Product')
plt.grid(linestyle = "-.")
plt.xticks(X_axis, X)
```

```
plt.xlabel("Groups")
plt.ylabel("Corr. Coefficient")
plt.title("Correlation Promotion Increase vs Nonprom
    Sales")
plt.legend()
plt.show()
...
fig = plt.figure(figsize = (6, 3))

plt.subplot(1,2,1)
X = ['Magnitude','Ratio']
Product = [products_corr_prom, products_corr_prom_ratio]
Stores = [stores_corr_prom_without91,
    stores_corr_prom_ratio]
X_axis = np.arange(len(X))
plt.bar(X_axis + 0.2, Stores, 0.4, label = 'Stores')
plt.bar(X_axis - 0.2, Product, 0.4, label = 'Product')
plt.grid(linestyle = "-.")
plt.xticks(X_axis, X)
plt.xlabel("Groups")
plt.ylabel("Corr. Coefficient")
plt.title("Correlation Promotion Increase(Without Store
    91)")
plt.legend()
plt.show()
...
reduced_rmse_stores    = 100*(
    RMSE_STORES_MAGNITUDE_NOT_CLUSTERED -
    RMSE_STORES_MAGNITUDE)/(
    RMSE_STORES_MAGNITUDE_NOT_CLUSTERED)
reduced_rmse_products = 100*(
    RMSE_PRODUCTS_MAGNITUDE_NOT_CLUSTERED -
    RMSE_PRODUCTS_MAGNITUDE)/(
    RMSE_PRODUCTS_MAGNITUDE_NOT_CLUSTERED)
print("For Stores Sale Forecasting, RMSE is reduced by
    %1.2f" % reduced_rmse_stores,"%")
print("For Products Sale Forecasting, RMSE is reduced by
    %1.2f" % reduced_rmse_products,"%")
...
fig = plt.figure(figsize = (6, 3))

plt.subplot(1,2,1)
plt.bar(["Stores","Products"], [reduced_rmse_stores,
    reduced_rmse_products])
plt.grid(linestyle = "-.")
plt.ylabel("Reduced RMSE(%)")
```

```
plt.title("Reduced RMSE by Strategy A")
plt.legend()
plt.show()
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

### Load Product Categories
...
df_product_categories = pd.read_csv('assignment4.1c.csv')
...
print(df_product_categories)
...
df_product_categories.iloc[[160, 163, 165, 182, 226, 227,
    228, 309]] ## products returned positive from
    negative
...
df_product_categories.iloc[[205, 207, 209, 210, 213, 218,
    219, 220, 221]] ## products affected more both in
    sale and in ratio
...
df_product_categories.iloc[[13, 193, 229, 231, 268, 291]]
    #products affected more in ratio
...
temp = df_product_categories.loc[(df_product_categories["
    ProductGroup1"]=='G')&(df_product_categories["
    ProductGroup2"]==4)]
G4   = temp["ProductCode"].to_numpy()
...
temp = df_product_categories.loc[(df_product_categories["
    ProductGroup1"]=='A')&(df_product_categories["
    ProductGroup2"]==5)]
A5   = temp["ProductCode"].to_numpy()
...
print(G4)
print(np.mean(difference_average_sale_product[G4-1]))
print(np.mean(difference_average_future_sale_product[G4
    -1]))
plt.plot(difference_average_sale_product)
plt.grid()

np.where(difference_average_sale_product>1)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

## These are genereally from Categories A5 AND G4. These
    two categories respond more than others.
...
print(np.mean(difference_average_sale_product))
```

```
print(np.mean(difference_average_sale_product[A5-1]))
print(np.mean(difference_average_future_sale_product[A5
    -1]))
...
plt.plot(average_weekly_sale_per_store)
...
fast_products = (np.array(fast_product_index)).reshape
    (-1,)
print(fast_products.shape)
temp = df_product_categories.iloc[fast_products]
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```