Name Gisanrin Oluwaseun

Email seungisanrin@gmail.com

GitHub github.com/seungisanrin

Quantium Data Analytics Internship - Task Two (2)

Loading Libraries and Datasets for the Analysis

Loading the necessary libraries

import pandas as pd

import seaborn as sns

import matplotlib.pyplot as plt

from statistics import stdev

Loading the cleaned and merged dataaset

df = pd.read_csv('QVI_data.csv')

df.shape

df.dtypes

df.describe()

df.head()

Converting the date column to a datetime object

df['DATE'] = pd.to_datetime(df['DATE'])

Extracting Stores with Transactions in every Month

len(df['STORE_NBR'].unique())

df['MONTH'] = df['DATE'].dt.strftime('%B')

df['YEAR_MONTH'] = df['DATE'].dt.strftime('%Y-%m')

Number of months with transactions per store

n_months_perstore = df.groupby('STORE_NBR')['YEAR_MONTH'].nunique().reset_index()

Filtering for stores that have transactions recorded for every month

full_obs = n_months_perstore[n_months_perstore['YEAR_MONTH'] == 12]

full_obs

The stores who don't have transactions for at least one month have been filtered, now to reflect on the dataset.

Storing the eligible stores for the analysis to a variable

store_df = df

Extracting the relevant informmation from the

monthly_revenue = store_df.groupby(['STORE_NBR','YEAR_MONTH'])['TOT_SALES'].sum().reset_index()

monthly_customers =
store_df.groupby(['STORE_NBR','YEAR_MONTH'])['LYLTY_CARD_NBR'].nunique().reset_index()

```
monthly_txn = store_df.groupby(['STORE_NBR','YEAR_MONTH'])['TXN_ID'].count().reset_index()

monthly_nchips = store_df.groupby(['STORE_NBR','YEAR_MONTH'])['PROD_QTY'].sum().reset_index()
# Aggregating all the necessary metrics for further analysis
store_agg = pd.concat([monthly_revenue, monthly_customers['LYLTY_CARD_NBR'],
                monthly_txn['TXN_ID'], monthly_nchips['PROD_QTY']],
                axis=1,
                ignore_index=True)

store_agg.columns = ['store_nbr','year_month','tot_sales','n_customers',
            'n_txns','n_chips']
# Computing more of the necessary metrics
store_agg['txn_per_customer'] = store_agg['n_txns']/store_agg['n_customers']

store_agg['chips_per_customer'] = store_agg['n_chips']/store_agg['n_customers']

store_agg['avgprice_per_unit'] = store_agg['tot_sales']/store_agg['n_chips']
# Dropping rows that are not necessary for the analysis
store_agg.drop(columns=['n_txns','n_chips'], inplace=True)
store_agg.head()
store_agg.info()
# Filtering the store aggregate data for pretrial metrics, before February 2019
pretrial_metric = store_agg[(store_agg['year_month'] < '2019-02') &
                    (store_agg['store_nbr'].isin(full_obs['STORE_NBR']))]
pretrial_metric.shape
# Function to calculate the correlation between a trial store and control stores
def calcCorrelation(metric_col,trial_store,df_x):
  '''
  Calculates the correlation coefficient between all the control stores and the
  specified trial store.

  Args:
    metric_col (str): The column metric on which the correlation coefficient should be
```

```
            calculated

    trial_store (int): The trial store number that is being compared with other control stores.
            Options are: 'tot_sales', 'n_customers', 'txn_per_customer',
            'chips_per_customer', 'avgprice_per_unit'.

    df (pd.DataFrame): The dataframe containing all metrics for both trial and control stores

Returns:
    Pandas DataFrama Object: Dataframe with three columns the trial store number,
    control store number, and the magnitude distance.
'''

# Creating an output dictionary to store the values generated
output = {'trial_store': [],'control_store':[],'corr_coef':[]}

# Extracting the specified metric column for the trial stores
trial_val = df_x[df_x['store_nbr'] == trial_store][metric_col].reset_index(drop=True)

# Creating a series for the store numbers to loop through
store_num = df_x['store_nbr'].unique()

# Looping through all the stores to generate the coefficient metric
for i in store_num:
    control_val = df_x[df_x['store_nbr'] == i][metric_col].reset_index(drop=True)
    output['trial_store'].append(trial_store)
    output['control_store'].append(i)
    output['corr_coef'].append(control_val.corr(trial_val))

# Saving the populated dictionary to a dataframe
output_df = pd.DataFrame(output)

return output_df
```

Calculating Magnitude Distance

1 - (Observed distance – minimum distance)/(Maximum distance – minimum distance)

```
# Function to calculate the magnitude distance between a trial store and control stores
def calcMagnitudeDistance(metric_col,trial_store,df_x):
    '''
    Calculates the magnitude distance between all the control stores and the
    specified trial store

    Args:
        metric_col (str): The column metric on which the magnitude distance should be calculated

        trial_store (int): The trial store number that is being compared with other control stores.
                    Options are: 'tot_sales', 'n_customers', 'txn_per_customer',
                    'chips_per_customer', 'avgprice_per_unit'.

        df (pd.DataFrame): The dataframe containing all metrics for both trial and control stores

    Returns:
        Pandas DataFrama Object: Dataframe with three columns the trial store number,
        control store number, and the magnitude distance.
    '''

    # Creating an output dictionary to store the values generated
    output = {'trial_store': [],'control_store':[],'mag_distance':[]}

    # Extracting the specified metric column for the trial stores
    trial_val = df_x[df_x['store_nbr'] == trial_store][metric_col].reset_index(drop=True)

    # Creating a series for the store numbers to loop through
    store_num = df_x['store_nbr'].unique()

    # Looping through all the stores to generate the coefficient metric
    for i in store_num:
```

```
        control_val = df_x[df_x['store_nbr'] == i][metric_col].reset_index(drop=True)
        # Defining the standard distance value
        standard = abs(control_val-trial_val)
        # Calculating the magnitude by using the provided distance formula and taking the mean
        # of the resulting series
        magnitude = (1 - ((standard-min(standard))/(max(standard)-min(standard)))).mean()
        # Saving to the dictionary variable
        output['trial_store'].append(trial_store)
        output['control_store'].append(i)
        output['mag_distance'].append(magnitude)

    # Saving the populated dictionary to a dataframe
    output_df = pd.DataFrame(output)

    return output_df
```

Store 77: Total Sales and Number of Customers Correlations and Magnitude Distance

```
trial_store = 77

corr_tot_sales = calcCorrelation('tot_sales',trial_store,pretrial_metric)
corr_n_customers = calcCorrelation('n_customers',trial_store,pretrial_metric)

mag_tot_sales = calcMagnitudeDistance('tot_sales',trial_store,pretrial_metric)
mag_n_customers = calcMagnitudeDistance('n_customers',trial_store,pretrial_metric)
sales77 = pd.concat([corr_tot_sales,mag_tot_sales['mag_distance']], axis=1)

sales77['sales_score'] = (sales77['corr_coef']*0.5) + (sales77['mag_distance']*0.5)
customer77 = pd.concat([corr_n_customers,mag_n_customers['mag_distance']], axis=1)

customer77['customer_score'] = (customer77['corr_coef']*0.5) + (customer77['mag_distance']*0.5)
customer77.head()
sales77.head()
# Merging sales and customer scores to one dataframe
score_control_77 = sales77.merge(customer77,on=['trial_store','control_store'])
```

```
# Cleaning up the column name for the merged dataframe
score_control_77.rename(columns={'corr_coef_x': 'sales_corr_coef', 'mag_distance_x': 'sales_mag_distance',
                                 'corr_coef_y': 'cust_corr_coef', 'mag_distance_y': 'cust_mag_distance'}, inplace=True)
score_control_77.head()
score_control_77['final_control_score'] = (score_control_77['sales_score'] * 0.5) +
(score_control_77['customer_score'] * 0.5)
# Extracting the control store ussing the score metric
control_store = score_control_77.sort_values(by='final_control_score',
                                             ascending=False).head(1)['control_store'].reset_index(drop=True).item()
# Assigning values to the initialised `store_type` column
store_agg['store_type'] = (
  store_agg['store_nbr']
  .map({trial_store: 'Trial', control_store: 'Control'})
  .fillna('Other Stores')
)
store_agg.head()
store_agg.columns
plt.figure(figsize=(7,5))
sns.lineplot(store_agg.loc[store_agg['year_month'] < '2019-02'],
        x='year_month',
        y='tot_sales',
        hue='store_type',
        err_style=None)
plt.title('Total Sales by Month')
plt.xlabel('Month of Operation')
plt.ylabel('Total Sales')
plt.legend(title = 'Store Type', loc='upper left', bbox_to_anchor=(1, 1), ncol=1)
plt.tight_layout()
plt.savefig('trial_77_monthly_sales.png')
plt.figure(figsize=(7,5))
sns.lineplot(store_agg.loc[store_agg['year_month'] < '2019-02'],
        x='year_month',
        y='n_customers',
```

```
        hue='store_type',
        err_style=None)
plt.title('Number of Customers by Month')
plt.xlabel('Month of Operation')
plt.ylabel('Number of Customers')
plt.legend(title = 'Store Type', loc='upper left', bbox_to_anchor=(1, 1), ncol=1)
plt.tight_layout()
plt.savefig('trial_77_n_customers.png')
```

From the visualisation above we can observe the similarity in sales performance and customer volume of stores 77 and 233 during the pretrial period, hence store 233 is serves as a control store suitable for the comparative analysis.

```
scaling_factor_sales = (pretrial_metric.loc[pretrial_metric['store_nbr'] == trial_store]['tot_sales'].sum()/
                pretrial_metric.loc[pretrial_metric['store_nbr'] == control_store]['tot_sales'].sum())
scaling_factor_sales
scaled_control_sales = store_agg.loc[store_agg['store_nbr'] == control_store]
scaled_control_sales = scaled_control_sales.assign(control_sales = (
  scaled_control_sales['tot_sales'] * scaling_factor_sales
))
percentDiff = scaled_control_sales[['year_month','control_sales']].reset_index(drop=True)
percentDiff = percentDiff.merge(right=store_agg[store_agg['store_nbr'] ==
trial_store][['year_month','tot_sales']].reset_index(drop=True).rename(columns={'tot_sales': 'trial_sales'}),
            on='year_month')
percentDiff['percentage_diff'] = (
  abs(percentDiff['control_sales'] - percentDiff['trial_sales'])/percentDiff['control_sales']
)
stdDev = stdev(percentDiff[(percentDiff['year_month'] < '2019-02')]['percentage_diff'])
degreesOfFreedom = 7
percentDiff['tValue'] = (percentDiff['percentage_diff'] - 0)/stdDev
percentDiff.head()
pastSales = (
  percentDiff[['year_month', 'trial_sales']]  # Extract columns
  .rename(columns={'trial_sales': 'totSales'})  # Rename column
  .assign(Store_type='Trial')  # Add new column
)
```

```
pastSales.head()
pastControlSales = (
   percentDiff[['year_month', 'control_sales']]  # Extract columns
   .rename(columns={'control_sales': 'totSales'})  # Rename column
   .assign(Store_type='Control')  # Add new column
)

pastControlSales.head()
pastSales_Control95 = (
   percentDiff[['year_month', 'control_sales']]  # Extract columns
   .assign(totSales=lambda df: df['control_sales'] * (1 + (stdDev * 2)))  # Compute totSales
   .drop(columns=['control_sales'])  # Drop the control_sales column
   .assign(Store_type='Control 95th % confidence interval')  # Add Store_type column
)

pastSales_Control95.head()
pastSales_Control5 = (
   percentDiff[['year_month', 'control_sales']]  # Extract columns
   .assign(totSales=lambda df: df['control_sales'] * (1 - (stdDev * 2)))  # Compute totSales
   .drop(columns=['control_sales'])  # Drop the control_sales column
   .assign(Store_type='Control 5th % confidence interval')  # Add Store_type column
)

pastSales_Control5.head()
trialAssessment = (pd.concat([pastSales,pastControlSales,pastSales_Control5,pastSales_Control95], axis=0)
            .sort_values('year_month')
            .reset_index(drop=True))

trialAssessment.head()
plt.figure(figsize=(10,6))
sns.lineplot(trialAssessment,
        x='year_month',
```

```
        y='totSales',
        hue='Store_type')
plt.xlabel('Month of Operation')
plt.ylabel('Total Sales')
plt.title('Total Sales by Month')
plt.ylim(bottom = 0)
plt.legend(title = 'Store Type')
plt.axvspan(xmin='2019-02', xmax='2019-04', color = '#d65f5f', alpha = 0.1)
plt.tight_layout()
plt.savefig('sales_77.png')
plt.show()
```

The results show that the trial in store 77 is evidently different from the control store during the trial period, as the trial store performance is significantly better than that of the control store in two out of the three month trial period.

```
scaling_factor_customer = (pretrial_metric.loc[pretrial_metric['store_nbr'] == trial_store]['n_customers'].sum()/
                    pretrial_metric.loc[pretrial_metric['store_nbr'] == control_store]['n_customers'].sum())
scaling_factor_customer
scaled_control_customer = store_agg.loc[store_agg['store_nbr'] == control_store]
scaled_control_customer = scaled_control_customer.assign(
  control_customer = (scaled_control_customer['n_customers'] * scaling_factor_customer)
)
scaled_control_customer.head()
trial_customer = store_agg[['year_month', 'store_nbr', 'n_customers']]
trial_customer = trial_customer.loc[trial_customer['store_nbr'] == trial_store]
trial_customer.drop(columns=['store_nbr'], inplace=True)
percentageDiff = (
  scaled_control_customer[['year_month','control_customer']]
  .merge(right = trial_customer, on = 'year_month')
  .rename(columns = {'n_customers' : 'trial_customers'})
  .assign(percDiff = lambda df: abs(df['control_customer'] - df['trial_customers'])/df['control_customer'])
)


percentageDiff.head()
```

```
stdDev = stdev(percentageDiff[percentageDiff['year_month'] < '2019-02']['percDiff'])

degreesOfFreedom = 7 # 8 months under observation, hence the degree of freedom is (8 - 1 = 7)

pastCustomers = store_agg[['year_month','n_customers','store_type']]

pastCustomers =
pastCustomers.loc[pastCustomers['store_type'].isin(['Trial','Control'])].reset_index(drop=True)


pastCustomers.head()

pastCustomers_Control95 = (

  pastCustomers[pastCustomers['store_type'] == 'Control']

  .assign(n_customers = lambda df: df['n_customers'] * (1 + stdDev * 2))

  .assign(store_type = 'Control 95th % confidence interval')

  .reset_index(drop=True)

)


pastCustomers_Control95.head()

pastCustomers_Control5 = (

  pastCustomers[pastCustomers['store_type'] == 'Control']

  .assign(n_customers = lambda df: df['n_customers'] * (1 - stdDev * 2))

  .assign(store_type = 'Control 5th % confidence interval')

  .reset_index(drop=True)

)


pastCustomers_Control5.head()

trialAssessment = (

  pd.concat([pastCustomers,pastCustomers_Control95,pastCustomers_Control5], axis=0)

  .sort_values('year_month', ascending=True)

  .reset_index(drop=True)

)


trialAssessment.head()

plt.figure(figsize=(10,6))

sns.lineplot(trialAssessment,

        x='year_month',
```

```
        y='n_customers',
        hue='store_type')
plt.xlabel('Month of Operation')
plt.ylabel('Number of Customers')
plt.title('Number of Customers by Month')
plt.ylim(bottom = 0)
plt.legend(title = 'Store Type')
plt.axvspan(xmin='2019-02', xmax='2019-04', color = '#d65f5f', alpha = 0.1)
plt.tight_layout()
plt.savefig('customer_77.png')
plt.show()
```

This customer volume analysis between stores 77 and 233 yielded similar results to the sales analysis carried out earlier. The trial store (77) performs significantly better than the control store in the last two months of the three month trial period.

Store 86: Total Sales and Number of Customers Correlations and Magnitude Distance

```
trial_store = 86

corr_tot_sales = calcCorrelation('tot_sales',trial_store,pretrial_metric)
corr_n_customers = calcCorrelation('n_customers',trial_store,pretrial_metric)

mag_tot_sales = calcMagnitudeDistance('tot_sales',trial_store,pretrial_metric)
mag_n_customers = calcMagnitudeDistance('n_customers',trial_store,pretrial_metric)
sales86 = pd.concat([corr_tot_sales,mag_tot_sales['mag_distance']], axis=1)

sales86['sales_score'] = (sales86['corr_coef']*0.5) + (sales86['mag_distance']*0.5)

sales86.head()
customer86 = pd.concat([corr_n_customers,mag_n_customers['mag_distance']], axis=1)

customer86['customer_score'] = (customer86['corr_coef']*0.5) + (customer86['mag_distance']*0.5)

customer86.head()
score_control_86 = sales86.merge(customer86,on=['trial_store','control_store'])
```

```
score_control_86.rename(columns={'corr_coef_x': 'sales_corr_coef', 'mag_distance_x': 'sales_mag_distance',
                        'corr_coef_y': 'cust_corr_coef', 'mag_distance_y': 'cust_mag_distance'}, inplace=True)
score_control_86.head()
score_control_86['final_control_score'] = (score_control_86['sales_score'] * 0.5) +
(score_control_86['customer_score'] * 0.5)
control_store = score_control_86.sort_values(by='final_control_score',
                                ascending=False).head(1)['control_store'].reset_index(drop=True).item()
control_store
store_agg['store_type'] = (store_agg['store_nbr']
                .map({trial_store: 'Trial', control_store: 'Control'})
                .fillna('Other Stores'))
plt.figure(figsize=(7,5))
sns.lineplot(store_agg.loc[store_agg['year_month'] < '2019-02'],
        x='year_month',
        y='tot_sales',
        hue='store_type',
        err_style=None)
plt.title('Total Sales by Month')
plt.xlabel('Month of Operation')
plt.ylabel('Total Sales')
plt.legend(title = 'Store Type', loc='upper left', bbox_to_anchor=(1, 1), ncol=1)
plt.tight_layout()
plt.savefig('trial_86_monthly_sales.png')
plt.show()
plt.figure(figsize=(7,5))
sns.lineplot(store_agg.loc[store_agg['year_month'] < '2019-02'],
        x='year_month',
        y='n_customers',
        hue='store_type',
        err_style=None)
plt.title('Number of Customers by Month')
plt.xlabel('Month of Operation')
plt.ylabel('Number of Customers')
```

```
plt.legend(title = 'Store Type', loc='upper left', bbox_to_anchor=(1, 1), ncol=1)

plt.tight_layout()

plt.savefig('trial_86_n_customers.png')

plt.show()
```

The visualisations above shows the similarity in sales performance and customer volume between the trial store (86) and the control store (155). Hence, the consequent suitability of store 155 as the control store for the comparative analysis with the trial store 86.

```
scaling_factor_sales = (pretrial_metric.loc[pretrial_metric['store_nbr'] == trial_store]['tot_sales'].sum()/
                        pretrial_metric.loc[pretrial_metric['store_nbr'] == control_store]['tot_sales'].sum())

scaling_factor_sales

scaled_control_sales = store_agg.loc[store_agg['store_nbr'] == control_store]

scaled_control_sales = scaled_control_sales.assign(
   control_sales = (scaled_control_sales['tot_sales'] * scaling_factor_sales)
)

percentDiff = scaled_control_sales[['year_month','control_sales']].reset_index(drop=True)

percentDiff = percentDiff.merge(right=store_agg[store_agg['store_nbr'] ==
trial_store][['year_month','tot_sales']].reset_index(drop=True).rename(columns={'tot_sales': 'trial_sales'}),
                                on='year_month')

percentDiff['percentage_diff'] = abs(percentDiff['control_sales'] -
percentDiff['trial_sales'])/percentDiff['control_sales']

stdDev = stdev(percentDiff[(percentDiff['year_month'] < '2019-02')]['percentage_diff'])

degreesOfFreedom = 7

percentDiff['tValue'] = (percentDiff['percentage_diff'] - 0)/stdDev

percentDiff.head()

pastSales = (
   percentDiff[['year_month', 'trial_sales']]  # Extract columns
   .rename(columns={'trial_sales': 'totSales'})  # Rename column
   .assign(Store_type='Trial')  # Add new column
)


pastSales.head()

pastControlSales = (
   percentDiff[['year_month', 'control_sales']]  # Extract columns
   .rename(columns={'control_sales': 'totSales'})  # Rename column
```

```
    .assign(Store_type='Control')  # Add new column
)

pastControlSales.head()
pastSales_Control95 = (
    percentDiff[['year_month', 'control_sales']]  # Extract columns
    .assign(totSales=lambda df: df['control_sales'] * (1 + (stdDev * 2)))  # Compute totSales
    .drop(columns=['control_sales'])  # Drop the control_sales column
    .assign(Store_type='Control 95th % confidence interval')  # Add Store_type column
)

pastSales_Control95.head()
pastSales_Control5 = (
    percentDiff[['year_month', 'control_sales']]  # Extract columns
    .assign(totSales=lambda df: df['control_sales'] * (1 - (stdDev * 2)))  # Compute totSales
    .drop(columns=['control_sales'])  # Drop the control_sales column
    .assign(Store_type='Control 5th % confidence interval')  # Add Store_type column
)

pastSales_Control5.head()
trialAssessment = (pd.concat([pastSales,pastControlSales,pastSales_Control5,pastSales_Control95], axis=0)
                .sort_values('year_month')
                .reset_index(drop=True))

trialAssessment.head()
plt.figure(figsize=(13,8))
sns.lineplot(trialAssessment,
             x='year_month',
             y='totSales',
             hue='Store_type')
plt.ylim(bottom = 0)
plt.xlabel('Month of Operation')
plt.ylabel('Total Sales')
```

```
plt.title('Total Sales by Month')
plt.legend(title = 'Store Type', bbox_to_anchor = (1,1))
plt.axvspan(xmin='2019-02', xmax='2019-04', color = '#d65f5f', alpha = 0.1)
plt.tight_layout()
plt.savefig('sales_86.png')
plt.show()
```

The results show that the trial in store 86 is not significantly different to its control store in the trial period as the trial store performance lies inside the 5% to 95% confidence interval of the control store in two of the three trial months.

```
scaling_factor_customer = (pretrial_metric.loc[pretrial_metric['store_nbr'] == trial_store]['n_customers'].sum()/
                           pretrial_metric.loc[pretrial_metric['store_nbr'] == control_store]['n_customers'].sum())
scaled_control_customer = store_agg.loc[store_agg['store_nbr'] == control_store]
scaled_control_customer = scaled_control_customer.assign(
  control_customer = (scaled_control_customer['n_customers'] * scaling_factor_customer)
)
trial_customer = store_agg[['year_month', 'store_nbr', 'n_customers']]
trial_customer = trial_customer.loc[trial_customer['store_nbr'] == trial_store]
trial_customer.drop(columns=['store_nbr'], inplace=True)
percentageDiff = (
  scaled_control_customer[['year_month','control_customer']]
  .merge(right = trial_customer, on = 'year_month')
  .rename(columns = {'n_customers' : 'trial_customers'})
  .assign(percDiff = lambda df: abs(df['control_customer'] - df['trial_customers'])/df['control_customer'])
)

percentageDiff.head()
stdDev = stdev(percentageDiff[percentageDiff['year_month'] < '2019-02']['percDiff'])
degreesOfFreedom = 7 # 8 months under observation, hence the degree of freedom is (8 - 1 = 7)
pastCustomers = store_agg[['year_month','n_customers','store_type']]
pastCustomers =
pastCustomers.loc[pastCustomers['store_type'].isin(['Trial','Control'])].reset_index(drop=True)

pastCustomers.head()
```

```
pastCustomers_Control95 = (
   pastCustomers[pastCustomers['store_type'] == 'Control']
   .assign(n_customers = lambda df: df['n_customers'] * (1 + stdDev * 2))
   .assign(store_type = 'Control 95th % confidence interval')
   .reset_index(drop=True)
)

pastCustomers_Control95.head()
pastCustomers_Control5 = (
   pastCustomers[pastCustomers['store_type'] == 'Control']
   .assign(n_customers = lambda df: df['n_customers'] * (1 - stdDev * 2))
   .assign(store_type = 'Control 5th % confidence interval')
   .reset_index(drop=True)
)

pastCustomers_Control5.head()
trialAssessment = (
   pd.concat([pastCustomers,pastCustomers_Control95,pastCustomers_Control5], axis=0)
   .sort_values('year_month', ascending=True)
   .reset_index(drop=True)
)

trialAssessment.head()
plt.figure(figsize=(10,6))
sns.lineplot(trialAssessment,
         x='year_month',
         y='n_customers',
         hue='store_type')
plt.xlabel('Month of Operation')
plt.ylabel('Number of Customers')
plt.title('Number of Customers by Month')
plt.ylim(bottom = 0)
plt.legend(title = 'Store Type')
```

```
plt.axvspan(xmin='2019-02', xmax='2019-04', color = '#d65f5f', alpha = 0.1)

plt.tight_layout()

plt.savefig('customer_86.png')

plt.show()
```

The result shows that there is a significant difference in the customer volume during the trial months, however since this not reflected in the sales analysis. We should check with the category manager if there were special deals on chips products resulting in lower prices, consequently less sales.

Store 88: Total Sales and Number of Customers Correlations and Magnitude Distance

```
trial_store = 88


corr_tot_sales = calcCorrelation('tot_sales',trial_store,pretrial_metric)

corr_n_customers = calcCorrelation('n_customers',trial_store,pretrial_metric)


mag_tot_sales = calcMagnitudeDistance('tot_sales',trial_store,pretrial_metric)

mag_n_customers = calcMagnitudeDistance('n_customers',trial_store,pretrial_metric)

sales88 = pd.concat([corr_tot_sales,mag_tot_sales['mag_distance']], axis=1)


sales88['sales_score'] = (sales88['corr_coef']*0.5) + (sales88['mag_distance']*0.5)


sales88.head()

customer88 = pd.concat([corr_n_customers,mag_n_customers['mag_distance']], axis=1)


customer88['customer_score'] = (customer88['corr_coef']*0.5) + (customer88['mag_distance']*0.5)


customer88.head()

score_control_88 = sales88.merge(customer88,on=['trial_store','control_store'])

score_control_88.rename(columns={'corr_coef_x': 'sales_corr_coef', 'mag_distance_x': 'sales_mag_distance',

                    'corr_coef_y': 'cust_corr_coef', 'mag_distance_y': 'cust_mag_distance'}, inplace=True)

score_control_88.head()

score_control_88['final_control_score'] = (score_control_88['sales_score'] * 0.5) +
(score_control_88['customer_score'] * 0.5)

score_control_88.sort_values('final_control_score', ascending=False).head()
```

After checking the significance of visualisations, store 237 was determined to be the most compatible and relevant to the trial store 88

```
control_store = 237

store_agg['store_type'] = store_agg['store_nbr'].map({trial_store: 'Trial', control_store: 'Control'}).fillna('Other Stores')

plt.figure(figsize=(7,5))

sns.lineplot(store_agg.loc[store_agg['year_month'] < '2019-02'],

        x='year_month',

        y='tot_sales',

        hue='store_type',

        err_style=None)

plt.title('Total Sales by Month')

plt.xlabel('Month of Operation')

plt.ylabel('Total Sales')

plt.legend(title = 'Store Type', loc='upper left', bbox_to_anchor=(1, 1), ncol=1)

plt.tight_layout()

plt.savefig('trial_88_monthly_sales.png')

plt.show()

plt.figure(figsize=(7,5))

sns.lineplot(store_agg.loc[store_agg['year_month'] < '2019-02'],

        x='year_month',

        y='n_customers',

        hue='store_type',

        err_style=None)

plt.title('Number of Customers by Month')

plt.xlabel('Month of Operation')

plt.ylabel('Number of Customers')

plt.legend(title = 'Store Type', loc='upper left', bbox_to_anchor=(1, 1), ncol=1)

plt.tight_layout()

plt.savefig('trial_88_n_customers.png')

plt.show()
```

It can be visually observed the significance of the similarity between the trial store (88) and the control store (237) in both sales and customer volume metrics. Hence, store 237 serves as a suitable store for the comparative analysis with the trial store (88)

```
scaling_factor_sales = (pretrial_metric.loc[pretrial_metric['store_nbr'] == trial_store]['tot_sales'].sum()/

                pretrial_metric.loc[pretrial_metric['store_nbr'] == control_store]['tot_sales'].sum())
```

```
scaling_factor_sales

scaled_control_sales = store_agg.loc[store_agg['store_nbr'] == control_store]

scaled_control_sales = scaled_control_sales.assign(

   control_sales = (scaled_control_sales['tot_sales'] * scaling_factor_sales)

)

percentDiff = scaled_control_sales[['year_month','control_sales']].reset_index(drop=True)

percentDiff = percentDiff.merge(right=store_agg[store_agg['store_nbr'] ==
trial_store][['year_month','tot_sales']].reset_index(drop=True).rename(columns={'tot_sales': 'trial_sales'}),

            on='year_month')

percentDiff['percentage_diff'] = abs(percentDiff['control_sales'] -
percentDiff['trial_sales'])/percentDiff['control_sales']

stdDev = stdev(percentDiff[(percentDiff['year_month'] < '2019-02')]['percentage_diff'])

degreesOfFreedom = 7

percentDiff['tValue'] = (percentDiff['percentage_diff'] - 0)/stdDev

percentDiff.head()

pastSales = (

   percentDiff[['year_month', 'trial_sales']]  # Extract columns

   .rename(columns={'trial_sales': 'totSales'})  # Rename column

   .assign(Store_type='Trial')  # Add new column

)


pastSales.head()

pastControlSales = (

   percentDiff[['year_month', 'control_sales']]  # Extract columns

   .rename(columns={'control_sales': 'totSales'})  # Rename column

   .assign(Store_type='Control')  # Add new column

)


pastControlSales.head()

pastSales_Control95 = (

   percentDiff[['year_month', 'control_sales']]  # Extract columns

   .assign(totSales=lambda df: df['control_sales'] * (1 + (stdDev * 2)))  # Compute totSales

   .drop(columns=['control_sales'])  # Drop the control_sales column
```

```
    .assign(Store_type='Control 95th % confidence interval')  # Add Store_type column
)

pastSales_Control95.head()
pastSales_Control5 = (
    percentDiff[['year_month', 'control_sales']]  # Extract columns
    .assign(totSales=lambda df: df['control_sales'] * (1 - (stdDev * 2)))  # Compute totSales
    .drop(columns=['control_sales'])  # Drop the control_sales column
    .assign(Store_type='Control 5th % confidence interval')  # Add Store_type column
)

pastSales_Control5.head()
trialAssessment = (pd.concat([pastSales,pastControlSales,pastSales_Control5,pastSales_Control95], axis=0)
                .sort_values('year_month')
                .reset_index(drop=True))

trialAssessment.head()
plt.figure(figsize=(13,8))
sns.lineplot(trialAssessment,
             x='year_month',
             y='totSales',
             hue='Store_type')
#plt.ylim(bottom = 0)
plt.xlabel('Month of Operation')
plt.ylabel('Total Sales')
plt.title('Total Sales by Month')
plt.ylim(bottom = 0)
plt.legend(title = 'Store Type', bbox_to_anchor = (1,1))
plt.axvspan(xmin='2019-02', xmax='2019-04', color = '#d65f5f', alpha = 0.1)
plt.tight_layout()
plt.savefig('sales_88.png')
plt.show()
```

The results show that there is significant difference in the sales perfomance between the trial and the control stores. The trial store performed evidently better than the control store in two months from the three month trial period.

```
scaling_factor_customer = (pretrial_metric.loc[pretrial_metric['store_nbr'] == trial_store]['n_customers'].sum()/
                           pretrial_metric.loc[pretrial_metric['store_nbr'] == control_store]['n_customers'].sum())

scaled_control_customer = store_agg.loc[store_agg['store_nbr'] == control_store]

scaled_control_customer = scaled_control_customer.assign(
  control_customer = (scaled_control_customer['n_customers'] * scaling_factor_customer)
)

trial_customer = store_agg[['year_month', 'store_nbr', 'n_customers']]

trial_customer = trial_customer.loc[trial_customer['store_nbr'] == trial_store]

trial_customer.drop(columns=['store_nbr'], inplace=True)

percentageDiff = (
  scaled_control_customer[['year_month','control_customer']]
  .merge(right = trial_customer, on = 'year_month')
  .rename(columns = {'n_customers' : 'trial_customers'})
  .assign(percDiff = lambda df: abs(df['control_customer'] - df['trial_customers'])/df['control_customer'])
)


percentageDiff.head()

stdDev = stdev(percentageDiff[percentageDiff['year_month'] < '2019-02']['percDiff'])

degreesOfFreedom = 7 # 8 months under observation, hence the degree of freedom is (8 - 1 = 7)

pastCustomers = store_agg[['year_month','n_customers','store_type']]

pastCustomers =
pastCustomers.loc[pastCustomers['store_type'].isin(['Trial','Control'])].reset_index(drop=True)

pastCustomers.head()

pastCustomers_Control95 = (
  pastCustomers[pastCustomers['store_type'] == 'Control']
  .assign(n_customers = lambda df: df['n_customers'] * (1 + stdDev * 2))
  .assign(store_type = 'Control 95th % confidence interval')
  .reset_index(drop=True)
)


pastCustomers_Control95.head()
```

```
pastCustomers_Control5 = (
   pastCustomers[pastCustomers['store_type'] == 'Control']
   .assign(n_customers = lambda df: df['n_customers'] * (1 - stdDev * 2))
   .assign(store_type = 'Control 5th % confidence interval')
   .reset_index(drop=True)
)

pastCustomers_Control5.head()
trialAssessment = (
   pd.concat([pastCustomers,pastCustomers_Control95,pastCustomers_Control5], axis=0)
   .sort_values('year_month', ascending=True)
   .reset_index(drop=True)
)

trialAssessment.head()
plt.figure(figsize=(10,6))
sns.lineplot(trialAssessment,
         x='year_month',
         y='n_customers',
         hue='store_type')
plt.xlabel('Month of Operation')
plt.ylabel('Number of Customers')
plt.title('Number of Customers by Month')
plt.ylim(bottom = 0)
plt.legend(title = 'Store Type')
plt.axvspan(xmin='2019-02', xmax='2019-04', color = '#d65f5f', alpha = 0.1)
plt.tight_layout()
plt.savefig('customers_88.png')
plt.show()
```

Total number of customers in the trial period for the trial store is significantly higher than the control store for two out of three months, which indicates a positive trial effect.

Conclusion

Based on the exploratory data analysis, trial stores 77 and 86 demonstrated a significant increase in the number of customers for at least two of the three months during the trial period. In contrast, while trial store 88 did show a notable rise in customer numbers, this increase was not statistically significant according to the analysis.