pip install pandas

Requirement already satisfied: pandas in c:\users\my pc\anaconda3\lib\site-packages (1.3.4)  
Requirement already satisfied: numpy>=1.17.3 in c:\users\my pc\anaconda3\lib\site-packages (from pandas) (1.20.3)  
Requirement already satisfied: python-dateutil>=2.7.3 in c:\users\my pc\anaconda3\lib\site-packages (from pandas) (2.8.2)  
Requirement already satisfied: pytz>=2017.3 in c:\users\my pc\anaconda3\lib\site-packages (from pandas) (2021.3)  
Requirement already satisfied: six>=1.5 in c:\users\my pc\anaconda3\lib\site-packages (from python-dateutil>=2.7.3->pandas) (1.16.0)  
Note: you may need to restart the kernel to use updated packages.

import pandas as pd

df = pd.read\_csv("C:\\Users\\My PC\\Desktop\\Insomnia\\Hay\_DA.csv")  
df.head()

Date [2023] Initial Ms\_P1 (lbs) Measured Ms\_P1(lbs) Initial Ms\_P2(lbs) \  
0 Sep.6 6.20 6.30 6.50   
1 Sep.8 6.20 6.31 6.51   
2 Sep.9 6.30 6.32 6.53   
3 Sep.13 6.30 6.32 6.54   
4 Sep.15 6.32 6.34 6.55   
  
 Measured Ms\_P2(lbs) Initial Ms\_P3(lbs) Measured Ms\_P3(lbs)   
0 7.01 7.60 7.80   
1 7.02 7.60 7.80   
2 7.04 7.60 7.82   
3 7.03 7.61 7.83   
4 7.06 7.61 7.83

df.info()

<class 'pandas.core.frame.DataFrame'>  
RangeIndex: 39 entries, 0 to 38  
Data columns (total 7 columns):  
 # Column Non-Null Count Dtype   
--- ------ -------------- -----   
 0 Date [2023] 39 non-null object   
 1 Initial Ms\_P1 (lbs) 39 non-null float64  
 2 Measured Ms\_P1(lbs) 39 non-null float64  
 3 Initial Ms\_P2(lbs) 39 non-null float64  
 4 Measured Ms\_P2(lbs) 39 non-null float64  
 5 Initial Ms\_P3(lbs) 39 non-null float64  
 6 Measured Ms\_P3(lbs) 39 non-null float64  
dtypes: float64(6), object(1)  
memory usage: 2.3+ KB

df.describe()

Initial Ms\_P1 (lbs) Measured Ms\_P1(lbs) Initial Ms\_P2(lbs) \  
count 39.000000 39.000000 39.000000   
mean 6.437436 6.526923 6.733077   
std 0.119312 0.167193 0.158996   
min 6.200000 6.300000 6.500000   
25% 6.360000 6.390000 6.600000   
50% 6.420000 6.490000 6.710000   
75% 6.530000 6.665000 6.860000   
max 6.650000 6.850000 7.040000   
  
 Measured Ms\_P2(lbs) Initial Ms\_P3(lbs) Measured Ms\_P3(lbs)   
count 39.000000 39.000000 39.000000   
mean 7.893846 7.738205 8.607949   
std 0.455583 0.120697 0.799270   
min 7.010000 7.600000 7.800000   
25% 7.915000 7.655000 7.895000   
50% 8.020000 7.720000 8.200000   
75% 8.210000 7.775000 9.350000   
max 8.380000 8.020000 9.890000

# Create a new column for the differences  
df['Difference\_Ms\_P1'] = df['Measured Ms\_P1(lbs)'] - df['Initial Ms\_P1 (lbs)']  
df['Difference\_Ms\_P2'] = df['Measured Ms\_P2(lbs)'] - df['Initial Ms\_P2(lbs)']  
df['Difference\_Ms\_P3'] = df['Measured Ms\_P3(lbs)'] - df['Initial Ms\_P3(lbs)']  
  
# Display the DataFrame with the new column  
print(df)

Date [2023] Initial Ms\_P1 (lbs) Measured Ms\_P1(lbs) Initial Ms\_P2(lbs) \  
0 Sep.6 6.20 6.30 6.50   
1 Sep.8 6.20 6.31 6.51   
2 Sep.9 6.30 6.32 6.53   
3 Sep.13 6.30 6.32 6.54   
4 Sep.15 6.32 6.34 6.55   
5 Sep.16 6.33 6.34 6.56   
6 Sep.20 6.34 6.36 6.57   
7 Sep.22 6.33 6.36 6.57   
8 Sep.23 6.34 6.37 6.58   
9 Sep.27 6.36 6.38 6.60   
10 Sep.29 6.37 6.40 6.60   
11 Sep.30 6.37 6.41 6.61   
12 Oct.4 6.36 6.40 6.62   
13 Oct.6 6.37 6.41 6.63   
14 Oct.7 6.38 6.43 6.65   
15 Oct.11 6.38 6.43 6.65   
16 Oct.13 6.39 6.45 6.67   
17 Oct.14 6.40 6.46 6.69   
18 Oct.18 6.42 6.47 6.70   
19 Oct.20 6.42 6.49 6.71   
20 Oct.21 6.43 6.50 6.73   
21 Oct.25 6.42 6.51 6.74   
22 Oct.27 6.44 6.56 6.74   
23 Oct.28 6.46 6.58 6.76   
24 Nov.1 6.45 6.59 6.78   
25 Nov.3 6.46 6.59 6.80   
26 Nov.4 6.48 6.60 6.81   
27 Nov.8 6.49 6.62 6.83   
28 Nov.10 6.52 6.65 6.85   
29 Nov.11 6.54 6.68 6.87   
30 Nov.15 6.56 6.70 6.88   
31 Nov.17 6.58 6.72 6.90   
32 Nov.18 6.60 6.73 6.92   
33 Nov.22 6.60 6.75 6.94   
34 Nov.24 6.61 6.76 6.96   
35 Nov.25 6.62 6.78 6.98   
36 Nov.29 6.63 6.80 7.00   
37 Dec.1 6.64 6.83 7.02   
38 Dec.2 6.65 6.85 7.04   
  
 Measured Ms\_P2(lbs) Initial Ms\_P3(lbs) Measured Ms\_P3(lbs) \  
0 7.01 7.60 7.80   
1 7.02 7.60 7.80   
2 7.04 7.60 7.82   
3 7.03 7.61 7.83   
4 7.06 7.61 7.83   
5 7.08 7.62 7.84   
6 7.09 7.62 7.85   
7 7.08 7.63 7.86   
8 7.91 7.64 7.87   
9 7.90 7.65 7.89   
10 7.92 7.66 7.90   
11 7.93 7.66 7.91   
12 7.92 7.66 7.92   
13 7.94 7.68 7.94   
14 7.99 7.69 7.96   
15 8.00 7.70 7.98   
16 8.00 7.71 7.99   
17 8.02 7.71 8.00   
18 8.00 7.71 8.10   
19 8.01 7.72 8.20   
20 8.03 7.73 8.40   
21 8.02 7.73 8.60   
22 8.04 7.74 8.80   
23 8.06 7.74 8.90   
24 8.09 7.75 8.90   
25 8.10 7.76 8.90   
26 8.14 7.76 9.00   
27 8.16 7.77 9.20   
28 8.19 7.77 9.30   
29 8.23 7.78 9.40   
30 8.25 7.78 9.50   
31 8.26 7.78 9.60   
32 8.28 7.79 9.80   
33 8.30 7.79 9.82   
34 8.32 8.00 9.83   
35 8.34 8.00 9.84   
36 8.35 8.01 9.86   
37 8.37 8.01 9.88   
38 8.38 8.02 9.89   
  
 Percentage\_Increase Difference\_Ms\_P1 Difference\_Ms\_P2 Difference\_Ms\_P3   
0 7.846154 0.10 0.51 0.20   
1 7.834101 0.11 0.51 0.20   
2 7.810107 0.02 0.51 0.22   
3 7.492355 0.02 0.49 0.22   
4 7.786260 0.02 0.51 0.22   
5 7.926829 0.01 0.52 0.22   
6 7.914764 0.02 0.52 0.23   
7 7.762557 0.03 0.51 0.23   
8 20.212766 0.03 1.33 0.23   
9 19.696970 0.02 1.30 0.24   
10 20.000000 0.03 1.32 0.24   
11 19.969743 0.04 1.32 0.25   
12 19.637462 0.04 1.30 0.26   
13 19.758673 0.04 1.31 0.26   
14 20.150376 0.05 1.34 0.27   
15 20.300752 0.05 1.35 0.28   
16 19.940030 0.06 1.33 0.28   
17 19.880419 0.06 1.33 0.29   
18 19.402985 0.05 1.30 0.39   
19 19.374069 0.07 1.30 0.48   
20 19.316493 0.07 1.30 0.67   
21 18.991098 0.09 1.28 0.87   
22 19.287834 0.12 1.30 1.06   
23 19.230769 0.12 1.30 1.16   
24 19.321534 0.14 1.31 1.15   
25 19.117647 0.13 1.30 1.14   
26 19.530103 0.12 1.33 1.24   
27 19.472914 0.13 1.33 1.43   
28 19.562044 0.13 1.34 1.53   
29 19.796215 0.14 1.36 1.62   
30 19.912791 0.14 1.37 1.72   
31 19.710145 0.14 1.36 1.82   
32 19.653179 0.13 1.36 2.01   
33 19.596542 0.15 1.36 2.03   
34 19.540230 0.15 1.36 1.83   
35 19.484241 0.16 1.36 1.84   
36 19.285714 0.17 1.35 1.85   
37 19.230769 0.19 1.35 1.87   
38 19.034091 0.20 1.34 1.87

df.head()

Date [2023] Initial Ms\_P1 (lbs) Measured Ms\_P1(lbs) Initial Ms\_P2(lbs) \  
0 Sep.6 6.20 6.30 6.50   
1 Sep.8 6.20 6.31 6.51   
2 Sep.9 6.30 6.32 6.53   
3 Sep.13 6.30 6.32 6.54   
4 Sep.15 6.32 6.34 6.55   
  
 Measured Ms\_P2(lbs) Initial Ms\_P3(lbs) Measured Ms\_P3(lbs) \  
0 7.01 7.60 7.80   
1 7.02 7.60 7.80   
2 7.04 7.60 7.82   
3 7.03 7.61 7.83   
4 7.06 7.61 7.83   
  
 Percentage\_Increase Difference\_Ms\_P1 Difference\_Ms\_P2 Difference\_Ms\_P3   
0 7.846154 0.10 0.51 0.20   
1 7.834101 0.11 0.51 0.20   
2 7.810107 0.02 0.51 0.22   
3 7.492355 0.02 0.49 0.22   
4 7.786260 0.02 0.51 0.22

df.describe()

Initial Ms\_P1 (lbs) Measured Ms\_P1(lbs) Initial Ms\_P2(lbs) \  
count 39.000000 39.000000 39.000000   
mean 6.437436 6.526923 6.733077   
std 0.119312 0.167193 0.158996   
min 6.200000 6.300000 6.500000   
25% 6.360000 6.390000 6.600000   
50% 6.420000 6.490000 6.710000   
75% 6.530000 6.665000 6.860000   
max 6.650000 6.850000 7.040000   
  
 Measured Ms\_P2(lbs) Initial Ms\_P3(lbs) Measured Ms\_P3(lbs) \  
count 39.000000 39.000000 39.000000   
mean 7.893846 7.738205 8.607949   
std 0.455583 0.120697 0.799270   
min 7.010000 7.600000 7.800000   
25% 7.915000 7.655000 7.895000   
50% 8.020000 7.720000 8.200000   
75% 8.210000 7.775000 9.350000   
max 8.380000 8.020000 9.890000   
  
 Percentage\_Increase Difference\_Ms\_P1 Difference\_Ms\_P2 \  
count 39.000000 39.000000 39.000000   
mean 17.173634 0.089487 1.160769   
std 4.835743 0.055438 0.335688   
min 7.492355 0.010000 0.490000   
25% 19.075869 0.040000 1.300000   
50% 19.472914 0.090000 1.320000   
75% 19.734409 0.135000 1.345000   
max 20.300752 0.200000 1.370000   
  
 Difference\_Ms\_P3   
count 39.000000   
mean 0.869744   
std 0.695234   
min 0.200000   
25% 0.240000   
50% 0.480000   
75% 1.575000   
max 2.030000

# Calculate the total percentage increase for 'Initial\_A' over time  
total\_percentage\_increase = ((df['Initial Ms\_P1 (lbs)'].iloc[-1] - df['Initial Ms\_P1 (lbs)'].iloc[0]) / df['Initial Ms\_P1 (lbs)'].iloc[0]) \* 100  
  
# Display the DataFrame with the calculated total percentage increase  
#print(df)  
print(f'Total Percentage Increase for Initial\_A: {total\_percentage\_increase:.2f}%')

Total Percentage Increase for Initial\_A: 7.26%

# Calculate the percentage increase for 'Initial\_A' over time  
df['Percentage\_Increase'] = ((df['Initial Ms\_P1 (lbs)'].shift(-1) - df['Initial Ms\_P1 (lbs)']) / df['Initial Ms\_P1 (lbs)']) \* 100  
  
# Calculate the average percentage increase  
average\_percentage\_increase = df['Percentage\_Increase'].mean()  
  
# Display the DataFrame with the calculated percentage increases  
#print(df)  
print(f'Average Percentage Increase for Initial Ms\_P1 (lbs) over time: {average\_percentage\_increase:.2f}%')

Average Percentage Increase for Initial Ms\_P1 (lbs) over time: 0.18%

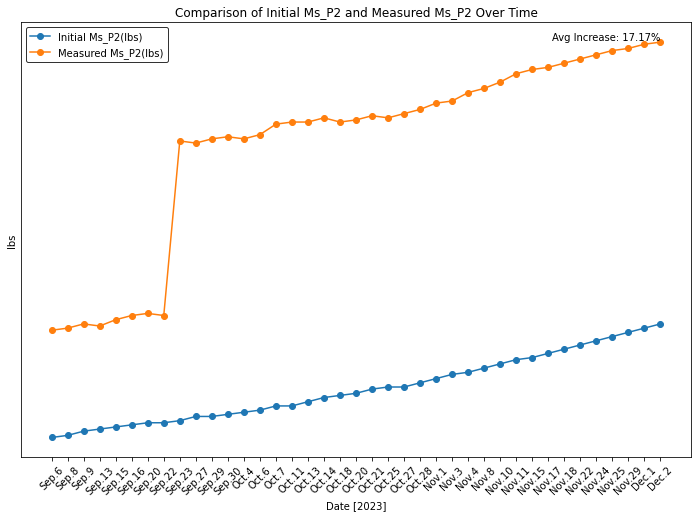
# Calculate the total percentage increase for 'Measured Ms\_P1' over time  
total\_percentage\_increase = ((df['Measured Ms\_P1(lbs)'].iloc[-1] - df['Measured Ms\_P1(lbs)'].iloc[0]) / df['Measured Ms\_P1(lbs)'].iloc[0]) \* 100  
  
# Display the DataFrame with the calculated total percentage increase  
#print(df)  
print(f'Total Percentage Increase for Measured Ms\_P1: {total\_percentage\_increase:.2f}%')

Total Percentage Increase for Measured Ms\_P1: 8.73%

# Calculate the percentage increase for 'Initial\_A' over time  
df['Percentage\_Increase'] = ((df['Measured Ms\_P1(lbs)'].shift(-1) - df['Measured Ms\_P1(lbs)']) / df['Measured Ms\_P1(lbs)']) \* 100  
  
# Calculate the average percentage increase  
average\_percentage\_increase = df['Percentage\_Increase'].mean()  
  
# Display the DataFrame with the calculated percentage increases  
#print(df)  
print(f'Average Percentage Increase for Measured Ms\_P1(lbs) over time: {average\_percentage\_increase:.2f}%')

Average Percentage Increase for Measured Ms\_P1(lbs) over time: 0.22%

import matplotlib.pyplot as plt  
  
# Calculate the percentage increase for 'Initial Ms\_P1 (lbs)' over time  
df['Percentage\_Increase'] = ((df['Measured Ms\_P2(lbs)'] - df['Initial Ms\_P2(lbs)']) / df['Initial Ms\_P2(lbs)']) \* 100  
  
# Calculate the average percentage increase  
average\_percentage\_increase = df['Percentage\_Increase'].mean()  
  
# Plotting the comparison over time  
plt.figure(figsize=(12, 8))  
plt.plot(df['Date [2023]'], df['Initial Ms\_P2(lbs)'], label='Initial Ms\_P2(lbs)', marker='o')  
plt.plot(df['Date [2023]'], df['Measured Ms\_P2(lbs)'], label='Measured Ms\_P2(lbs)', marker='o')  
  
# Display the average percentage increase as text on the plot  
plt.text(df['Date [2023]'].iloc[-1], df['Measured Ms\_P2(lbs)'].iloc[-1], f'Avg Increase: {average\_percentage\_increase:.2f}%', ha='right', va='bottom')  
  
# Rotate x-axis labels for better readability  
plt.xticks(rotation=45)  
  
# Add a gradient legend  
legend = plt.legend(loc='upper left', frameon=True)  
frame = legend.get\_frame()  
frame.set\_facecolor('white') # Adjust the face color as needed  
frame.set\_edgecolor('black') # Adjust the edge color as needed  
  
# Remove y-axis tick labels  
plt.yticks([])  
  
plt.title('Comparison of Initial Ms\_P2 and Measured Ms\_P2 Over Time')  
plt.xlabel('Date [2023]')  
plt.ylabel('Ibs')  
plt.grid(False)  
plt.show()



# Calculate the total percentage increase for 'Initial\_A' over time  
total\_percentage\_increase = ((df['Initial Ms\_P2(lbs)'].iloc[-1] - df['Initial Ms\_P2(lbs)'].iloc[0]) / df['Initial Ms\_P2(lbs)'].iloc[0]) \* 100  
  
# Display the DataFrame with the calculated total percentage increase  
#print(df)  
print(f'Total Percentage Increase for Initial Ms\_P2(lbs): {total\_percentage\_increase:.2f}%')

Total Percentage Increase for Initial Ms\_P2(lbs): 8.31%

# Calculate the percentage increase for 'Initial\_A' over time  
df['Percentage\_Increase'] = ((df['Initial Ms\_P2(lbs)'].shift(-1) - df['Initial Ms\_P2(lbs)']) / df['Initial Ms\_P2(lbs)']) \* 100  
  
# Calculate the average percentage increase  
average\_percentage\_increase = df['Percentage\_Increase'].mean()  
  
# Display the DataFrame with the calculated percentage increases  
#print(df)  
print(f'Average Percentage Increase for Initial Ms\_P2(lbs) (lbs) over time: {average\_percentage\_increase:.2f}%')

Average Percentage Increase for Initial Ms\_P2(lbs) (lbs) over time: 0.21%

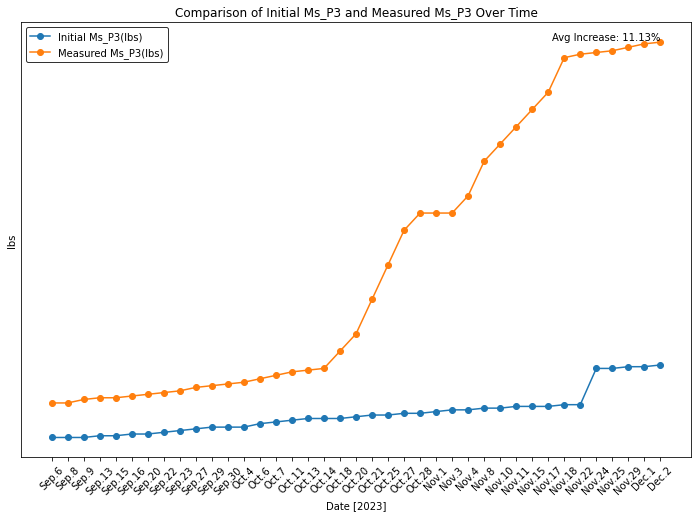
# Calculate the total percentage increase for 'Measured Ms\_P1' over time  
total\_percentage\_increase = ((df['Measured Ms\_P2(lbs)'].iloc[-1] - df['Measured Ms\_P2(lbs)'].iloc[0]) / df['Measured Ms\_P2(lbs)'].iloc[0]) \* 100  
  
# Display the DataFrame with the calculated total percentage increase  
#print(df)  
print(f'Total Percentage Increase for Measured Ms\_P2: {total\_percentage\_increase:.2f}%')

Total Percentage Increase for Measured Ms\_P2: 19.54%

# Calculate the percentage increase for 'Initial\_A' over time  
df['Percentage\_Increase'] = ((df['Measured Ms\_P2(lbs)'].shift(-1) - df['Measured Ms\_P2(lbs)']) / df['Measured Ms\_P2(lbs)']) \* 100  
  
# Calculate the average percentage increase  
average\_percentage\_increase = df['Percentage\_Increase'].mean()  
  
# Display the DataFrame with the calculated percentage increases  
#print(df)  
print(f'Average Percentage Increase for Measured Ms\_P2(lbs) over time: {average\_percentage\_increase:.2f}%')

Average Percentage Increase for Measured Ms\_P2(lbs) over time: 0.49%

import matplotlib.pyplot as plt  
  
# Calculate the percentage increase for 'Initial Ms\_P1 (lbs)' over time  
df['Percentage\_Increase'] = ((df['Measured Ms\_P3(lbs)'] - df['Initial Ms\_P3(lbs)']) / df['Initial Ms\_P3(lbs)']) \* 100  
  
# Calculate the average percentage increase  
average\_percentage\_increase = df['Percentage\_Increase'].mean()  
  
# Plotting the comparison over time  
plt.figure(figsize=(12, 8))  
plt.plot(df['Date [2023]'], df['Initial Ms\_P3(lbs)'], label='Initial Ms\_P3(lbs)', marker='o')  
plt.plot(df['Date [2023]'], df['Measured Ms\_P3(lbs)'], label='Measured Ms\_P3(lbs)', marker='o')  
  
# Display the average percentage increase as text on the plot  
plt.text(df['Date [2023]'].iloc[-1], df['Measured Ms\_P3(lbs)'].iloc[-1], f'Avg Increase: {average\_percentage\_increase:.2f}%', ha='right', va='bottom')  
  
# Rotate x-axis labels for better readability  
plt.xticks(rotation=45)  
  
# Add a gradient legend  
legend = plt.legend(loc='upper left', frameon=True)  
frame = legend.get\_frame()  
frame.set\_facecolor('white') # Adjust the face color as needed  
frame.set\_edgecolor('black') # Adjust the edge color as needed  
  
# Remove y-axis tick labels  
plt.yticks([])  
  
plt.title('Comparison of Initial Ms\_P3 and Measured Ms\_P3 Over Time')  
plt.xlabel('Date [2023]')  
plt.ylabel('Ibs')  
plt.grid(False)  
plt.show()



# Calculate the total percentage increase for 'Initial\_A' over time  
total\_percentage\_increase = ((df['Initial Ms\_P3(lbs)'].iloc[-1] - df['Initial Ms\_P3(lbs)'].iloc[0]) / df['Initial Ms\_P3(lbs)'].iloc[0]) \* 100  
  
# Display the DataFrame with the calculated total percentage increase  
#print(df)  
print(f'Total Percentage Increase for Initial Ms\_P3(lbs): {total\_percentage\_increase:.2f}%')

Total Percentage Increase for Initial Ms\_P3(lbs): 5.53%

# Calculate the percentage increase for 'Initial\_A' over time  
df['Percentage\_Increase'] = ((df['Initial Ms\_P3(lbs)'].shift(-1) - df['Initial Ms\_P3(lbs)']) / df['Initial Ms\_P3(lbs)']) \* 100  
  
# Calculate the average percentage increase  
average\_percentage\_increase = df['Percentage\_Increase'].mean()  
  
# Display the DataFrame with the calculated percentage increases  
#print(df)  
print(f'Average Percentage Increase for Initial Ms\_P3(lbs) (lbs) over time: {average\_percentage\_increase:.2f}%')

Average Percentage Increase for Initial Ms\_P3(lbs) (lbs) over time: 0.14%

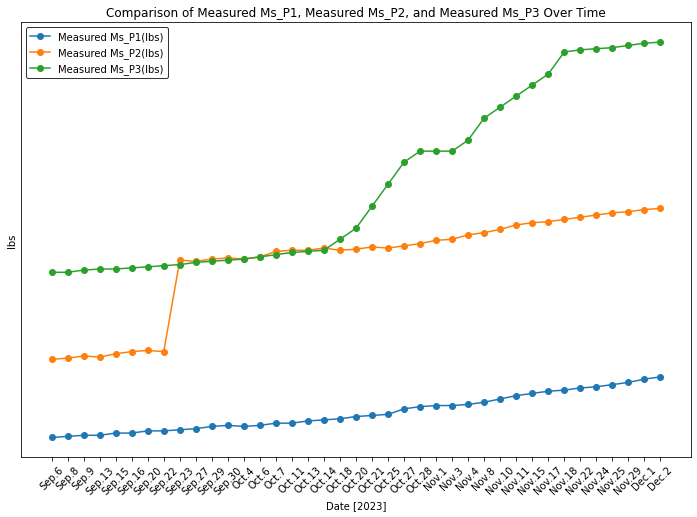
# Calculate the total percentage increase for 'Measured Ms\_P1' over time  
total\_percentage\_increase = ((df['Measured Ms\_P3(lbs)'].iloc[-1] - df['Measured Ms\_P3(lbs)'].iloc[0]) / df['Measured Ms\_P3(lbs)'].iloc[0]) \* 100  
  
# Display the DataFrame with the calculated total percentage increase  
#print(df)  
print(f'Total Percentage Increase for Measured Ms\_P3: {total\_percentage\_increase:.2f}%')

Total Percentage Increase for Measured Ms\_P3: 26.79%

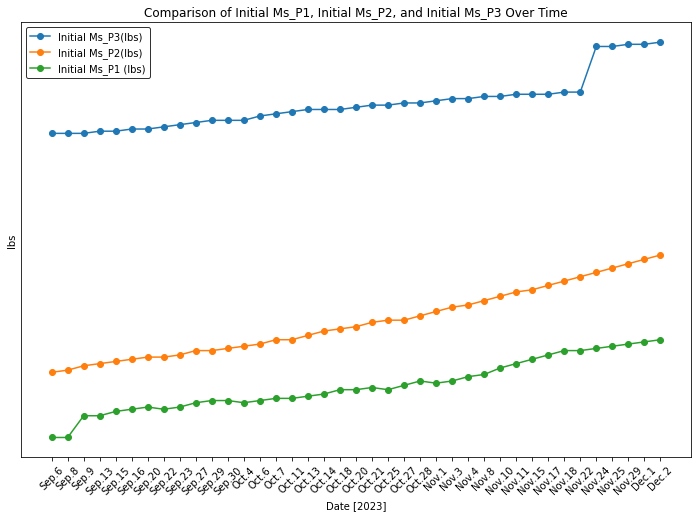
# Calculate the AVG\_percentage increase for 'Measured\_A' over time  
df['Percentage\_Increase'] = ((df['Measured Ms\_P3(lbs)'].shift(-1) - df['Measured Ms\_P3(lbs)']) / df['Measured Ms\_P3(lbs)']) \* 100  
  
# Calculate the average percentage increase  
average\_percentage\_increase = df['Percentage\_Increase'].mean()  
  
# Display the DataFrame with the calculated percentage increases  
#print(df)  
print(f'Average Percentage Increase for Measured Ms\_P3(lbs) over time: {average\_percentage\_increase:.2f}%')

Average Percentage Increase for Measured Ms\_P3(lbs) over time: 0.63%

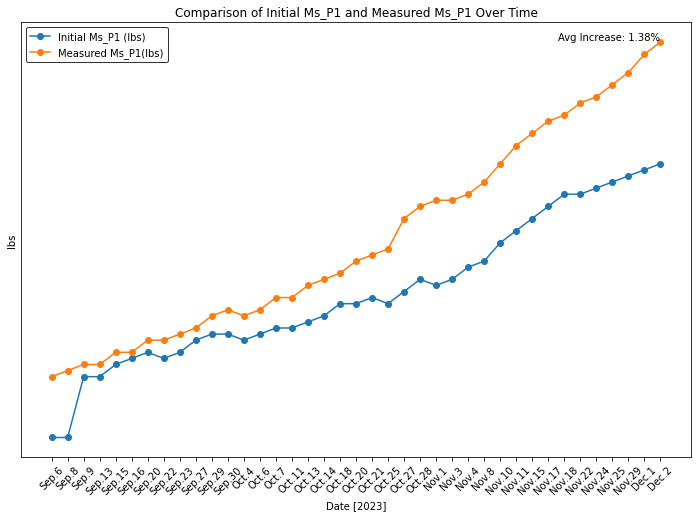
import matplotlib.pyplot as plt  
  
# Plotting the comparison over time  
plt.figure(figsize=(12, 8))  
plt.plot(df['Date [2023]'], df['Measured Ms\_P1(lbs)'], label='Measured Ms\_P1(lbs)', marker='o')  
plt.plot(df['Date [2023]'], df['Measured Ms\_P2(lbs)'], label='Measured Ms\_P2(lbs)', marker='o')  
plt.plot(df['Date [2023]'], df['Measured Ms\_P3(lbs)'], label='Measured Ms\_P3(lbs)', marker='o')  
  
# Rotate x-axis labels for better readability  
plt.xticks(rotation=45)  
  
# Add a gradient legend  
legend = plt.legend(loc='upper left', frameon=True)  
frame = legend.get\_frame()  
frame.set\_facecolor('white') # Adjust the face color as needed  
frame.set\_edgecolor('black') # Adjust the edge color as needed  
  
# Remove y-axis tick labels  
plt.yticks([])  
  
plt.title('Comparison of Measured Ms\_P1, Measured Ms\_P2, and Measured Ms\_P3 Over Time')  
plt.xlabel('Date [2023]')  
plt.ylabel('Ibs')  
plt.grid(False)  
plt.show()



import matplotlib.pyplot as plt  
  
# Plotting the comparison over time  
plt.figure(figsize=(12, 8))  
plt.plot(df['Date [2023]'], df['Initial Ms\_P3(lbs)'], label='Initial Ms\_P3(lbs)', marker='o')  
plt.plot(df['Date [2023]'], df['Initial Ms\_P2(lbs)'], label='Initial Ms\_P2(lbs)', marker='o')  
plt.plot(df['Date [2023]'], df['Initial Ms\_P1 (lbs)'], label='Initial Ms\_P1 (lbs)', marker='o')  
  
# Rotate x-axis labels for better readability  
plt.xticks(rotation=45)  
  
# Add a gradient legend  
legend = plt.legend(loc='upper left', frameon=True)  
frame = legend.get\_frame()  
frame.set\_facecolor('white') # Adjust the face color as needed  
frame.set\_edgecolor('black') # Adjust the edge color as needed  
  
# Remove y-axis tick labels  
plt.yticks([])  
  
plt.title('Comparison of Initial Ms\_P1, Initial Ms\_P2, and Initial Ms\_P3 Over Time')  
plt.xlabel('Date [2023]')  
plt.ylabel('Ibs')  
plt.grid(False)  
plt.show()



# Calculate the percentage increase for 'Initial Ms\_P1 (lbs)' over time  
df['Percentage\_Increase'] = ((df['Measured Ms\_P1(lbs)'] - df['Initial Ms\_P1 (lbs)']) / df['Initial Ms\_P1 (lbs)']) \* 100  
  
# Calculate the average percentage increase  
average\_percentage\_increase = df['Percentage\_Increase'].mean()  
  
# Plotting the comparison over time  
plt.figure(figsize=(12, 8))  
plt.plot(df['Date [2023]'], df['Initial Ms\_P1 (lbs)'], label='Initial Ms\_P1 (lbs)', marker='o')  
plt.plot(df['Date [2023]'], df['Measured Ms\_P1(lbs)'], label='Measured Ms\_P1(lbs)', marker='o')  
  
# Display the average percentage increase as text on the plot  
plt.text(df['Date [2023]'].iloc[-1], df['Measured Ms\_P1(lbs)'].iloc[-1], f'Avg Increase: {average\_percentage\_increase:.2f}%', ha='right', va='bottom')  
  
# Rotate x-axis labels for better readability  
plt.xticks(rotation=45)  
  
# Add a gradient legend  
legend = plt.legend(loc='upper left', frameon=True)  
frame = legend.get\_frame()  
frame.set\_facecolor('white') # Adjust the face color as needed  
frame.set\_edgecolor('black') # Adjust the edge color as needed  
  
# Remove y-axis tick labels  
plt.yticks([])  
  
plt.title('Comparison of Initial Ms\_P1 and Measured Ms\_P1 Over Time')  
plt.xlabel('Date [2023]')  
plt.ylabel('Ibs')  
plt.grid(False)  
plt.show()



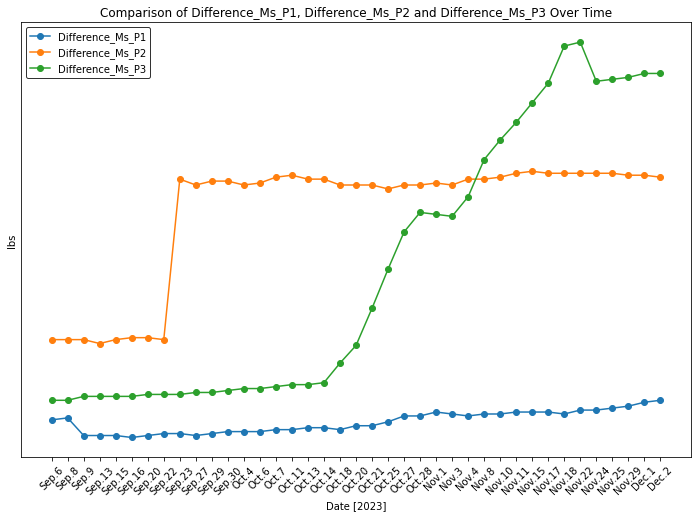
from scipy.stats import ttest\_ind  
  
# Extract the two series to compare  
data1 = df['Measured Ms\_P1(lbs)']  
data2 = df['Measured Ms\_P3(lbs)']  
  
# Perform an independent two-sample t-test  
t\_statistic, p\_value = ttest\_ind(data1, data2)  
  
print(t\_statistic, p\_value)  
  
# Check if the p-value is below the significance level (e.g., 0.05)  
significance\_level = 0.05  
if p\_value < significance\_level:  
 print(f"There is a significant difference between 'Measured Ms\_P1(lbs)' and 'Measured Ms\_P3(lbs)' (p-value: {p\_value:.4f})")  
else:  
 print(f"No significant difference between 'Measured Ms\_P1(lbs)' and 'Measured Ms\_P3(lbs)' (p-value: {p\_value:.4f})")

-15.915361585478253 6.575399025889357e-26  
There is a significant difference between 'Measured Ms\_P1(lbs)' and 'Measured Ms\_P3(lbs)' (p-value: 0.0000)

from scipy.stats import ttest\_ind  
  
# Extract the two series to compare  
data1 = df['Difference\_Ms\_P1']  
data2 = df['Difference\_Ms\_P3']  
  
# Perform an independent two-sample t-test  
t\_statistic, p\_value = ttest\_ind(data1, data2)  
  
print(t\_statistic, p\_value)  
  
# Check if the p-value is below the significance level (e.g., 0.05)  
significance\_level = 0.05  
if p\_value < significance\_level:  
 print(f"There is a significant difference between 'Measured Ms\_P1(lbs)' and 'Measured Ms\_P3(lbs)' (p-value: {p\_value:.4f})")  
else:  
 print(f"No significant difference between 'Measured Ms\_P1(lbs)' and 'Measured Ms\_P3(lbs)' (p-value: {p\_value:.4f})")

-6.9865447167369465 9.308654292119671e-10  
There is a significant difference between 'Measured Ms\_P1(lbs)' and 'Measured Ms\_P3(lbs)' (p-value: 0.0000)

import matplotlib.pyplot as plt  
  
# Plotting the comparison over time  
plt.figure(figsize=(12, 8))  
plt.plot(df['Date [2023]'], df['Difference\_Ms\_P1'], label='Difference\_Ms\_P1', marker='o')  
plt.plot(df['Date [2023]'], df['Difference\_Ms\_P2'], label='Difference\_Ms\_P2', marker='o')  
plt.plot(df['Date [2023]'], df['Difference\_Ms\_P3'], label='Difference\_Ms\_P3', marker='o')  
  
# Rotate x-axis labels for better readability  
plt.xticks(rotation=45)  
  
# Add a gradient legend  
legend = plt.legend(loc='upper left', frameon=True)  
frame = legend.get\_frame()  
frame.set\_facecolor('white') # Adjust the face color as needed  
frame.set\_edgecolor('black') # Adjust the edge color as needed  
  
# Remove y-axis tick labels  
plt.yticks([])  
  
plt.title('Comparison of Difference\_Ms\_P1, Difference\_Ms\_P2 and Difference\_Ms\_P3 Over Time')  
plt.xlabel('Date [2023]')  
plt.ylabel('Ibs')  
plt.grid(False)  
plt.show()



# Calculate the AVG\_percentage increase for 'Measured\_A' over time  
df['Percentage\_Increase'] = ((df['Difference\_Ms\_P3'].shift(-1) - df['Difference\_Ms\_P3']) / df['Difference\_Ms\_P3']) \* 100  
  
# Calculate the average percentage increase  
average\_percentage\_increase = df['Percentage\_Increase'].mean()  
  
# Display the DataFrame with the calculated percentage increases  
#print(df)  
print(f'Average Percentage Increase for Difference\_Ms\_P3 over time: {average\_percentage\_increase:.2f}%')

Average Percentage Increase for Difference\_Ms\_P3 over time: 6.51%

# Calculate the AVG\_percentage increase for 'Measured\_A' over time  
df['Percentage\_Increase'] = ((df['Difference\_Ms\_P2'].shift(-1) - df['Difference\_Ms\_P2']) / df['Difference\_Ms\_P2']) \* 100  
  
# Calculate the average percentage increase  
average\_percentage\_increase = df['Percentage\_Increase'].mean()  
  
# Display the DataFrame with the calculated percentage increases  
#print(df)  
print(f'Average Percentage Increase for Difference\_Ms\_P2 over time: {average\_percentage\_increase:.2f}%')

Average Percentage Increase for Difference\_Ms\_P2 over time: 4.26%

# Create a new column for the differences  
df['Difference\_Ms\_P1'] = df['Measured Ms\_P1(lbs)'] - df['Initial Ms\_P1 (lbs)']  
  
# Calculate the total increase and the average rate of increase  
total\_increase = df['Difference\_Ms\_P1'].iloc[-1] - df['Difference\_Ms\_P1'].iloc[0]  
  
rate\_of\_increase = (total\_increase / df['Difference\_Ms\_P1'].iloc[0])  
  
average\_rate\_of\_increase\_1 = (total\_increase / df['Difference\_Ms\_P1'].iloc[0]) \* 100 / (len(df) - 1)  
  
# Calculate the percentage increase for ' Ms\_P1 (lbs)' over time  
df['Percentage\_Increase'] = ((df['Measured Ms\_P1(lbs)'] - df['Initial Ms\_P1 (lbs)']) / df['Initial Ms\_P1 (lbs)']) \* 100  
  
# Calculate the average percentage increase overtime  
average\_percentage\_increase\_2 = df['Percentage\_Increase'].mean()  
  
#Display the total increase of Ms\_P1  
print(f'Total Increase in Difference\_Ms\_P1: {total\_increase:.2f}')  
  
# Display the total rate of increase  
print(f'Total Rate of Increase in Difference\_Ms\_P1: {rate\_of\_increase:.2f}')  
  
# Display the average rate of increase per unit period  
print(f'Average Rate of Increase in Difference\_Ms\_P1 per unit period: {average\_rate\_of\_increase\_1:.2f}%')  
  
# Display the average rate of increase over time  
print(f'Average Rate of Increase in Difference\_Ms\_P1 per over time: {average\_percentage\_increase\_2:.2f}%')

Total Increase in Difference\_Ms\_P1: 0.10  
Total Rate of Increase in Difference\_Ms\_P1: 1.00  
Average Rate of Increase in Difference\_Ms\_P1 per unit period: 2.63%  
Average Rate of Increase in Difference\_Ms\_P1 per over time: 1.38%

# Create a new column for the differences  
df['Difference\_Ms\_P2'] = df['Measured Ms\_P2(lbs)'] - df['Initial Ms\_P2(lbs)']  
  
# Calculate the total increase and the average rate of increase  
total\_increase = df['Difference\_Ms\_P2'].iloc[-1] - df['Difference\_Ms\_P2'].iloc[0]  
  
rate\_of\_increase = (total\_increase / df['Difference\_Ms\_P2'].iloc[0])  
  
average\_rate\_of\_increase\_1 = (total\_increase / df['Difference\_Ms\_P2'].iloc[0]) \* 100 / (len(df) - 1)  
  
# Calculate the percentage increase for 'Initial Ms\_P2 (lbs)' over time  
df['Percentage\_Increase'] = ((df['Measured Ms\_P2(lbs)'] - df['Initial Ms\_P2(lbs)']) / df['Initial Ms\_P2(lbs)']) \* 100  
  
# Calculate the average percentage increase overtime  
average\_percentage\_increase\_2 = df['Percentage\_Increase'].mean()  
  
#Display the total increase of Ms\_P2  
print(f'Total Increase in Difference\_Ms\_P2: {total\_increase:.2f}')  
  
# Display the total rate of increase  
print(f'Total Rate of Increase in Difference\_Ms\_P2: {rate\_of\_increase:.2f}')  
  
# Display the average rate of increase per unit period  
print(f'Average Rate of Increase in Difference\_Ms\_P2 per unit period: {average\_rate\_of\_increase\_1:.2f}%')  
  
# Display the average rate of increase over time  
print(f'Average Rate of Increase in Difference\_Ms\_P2 per over time: {average\_percentage\_increase\_2:.2f}%')

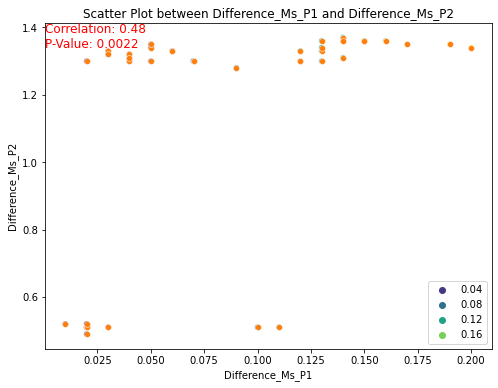
Total Increase in Difference\_Ms\_P2: 0.83  
Total Rate of Increase in Difference\_Ms\_P2: 1.63  
Average Rate of Increase in Difference\_Ms\_P2 per unit period: 4.28%  
Average Rate of Increase in Difference\_Ms\_P2 per over time: 17.17%

# Create a new column for the differences  
df['Difference\_Ms\_P3'] = df['Measured Ms\_P3(lbs)'] - df['Initial Ms\_P3(lbs)']  
  
# Calculate the total increase and the average rate of increase  
total\_increase = df['Difference\_Ms\_P3'].iloc[-1] - df['Difference\_Ms\_P3'].iloc[0]  
  
rate\_of\_increase = (total\_increase / df['Difference\_Ms\_P3'].iloc[0])  
  
average\_rate\_of\_increase\_1 = (total\_increase / df['Difference\_Ms\_P3'].iloc[0]) \* 100 / (len(df) - 1)  
  
# Calculate the percentage increase for 'Initial Ms\_P1 (lbs)' over time  
df['Percentage\_Increase'] = ((df['Measured Ms\_P3(lbs)'] - df['Initial Ms\_P3(lbs)']) / df['Initial Ms\_P3(lbs)']) \* 100  
  
# Calculate the average percentage increase overtime  
average\_percentage\_increase\_2 = df['Percentage\_Increase'].mean()  
  
#Display the total increase of Ms\_P3  
print(f'Total Increase in Difference\_Ms\_P3: {total\_increase:.2f}')  
  
# Display the total rate of increase  
print(f'Total Rate of Increase in Difference\_Ms\_P3: {rate\_of\_increase:.2f}')  
  
# Display the average rate of increase per unit period  
print(f'Average Rate of Increase in Difference\_Ms\_P3 per unit period: {average\_rate\_of\_increase\_1:.2f}%')  
  
# Display the average rate of increase over time  
print(f'Average Rate of Increase in Difference\_Ms\_P3 per over time: {average\_percentage\_increase\_2:.2f}%')

Total Increase in Difference\_Ms\_P3: 1.67  
Total Rate of Increase in Difference\_Ms\_P3: 8.35  
Average Rate of Increase in Difference\_Ms\_P3 per unit period: 21.97%  
Average Rate of Increase in Difference\_Ms\_P3 per over time: 11.13%

from scipy.stats import pearsonr  
import seaborn as sns  
  
  
# Calculate Pearson correlation coefficient  
correlation\_coefficient, p\_value = pearsonr(df['Difference\_Ms\_P1'], df['Difference\_Ms\_P2'])  
  
# Display the correlation coefficient and p-value  
print(f"Pearson Correlation Coefficient: {correlation\_coefficient}")  
print(f"P-Value: {p\_value}")  
  
# Plot a scatter plot to visualize the relationship  
plt.figure(figsize=(8, 6))  
  
# Scatter plot with different colors for each variable  
sns.scatterplot(x='Difference\_Ms\_P1', y='Difference\_Ms\_P2', data=df, hue=df['Difference\_Ms\_P1'], palette='viridis')  
  
# Annotate the plot with correlation coefficient and p-value  
plt.text(plt.xlim()[0], plt.ylim()[1], f'Correlation: {correlation\_coefficient:.2f}\nP-Value: {p\_value:.4f}',  
 horizontalalignment='left', verticalalignment='top', fontsize=12, color='red')  
  
sns.scatterplot(x='Difference\_Ms\_P1', y='Difference\_Ms\_P2', data=df)  
plt.title('Scatter Plot between Difference\_Ms\_P1 and Difference\_Ms\_P2')  
plt.show()  
  
# Check significance based on the p-value  
alpha = 0.05 # significance level  
if p\_value < alpha:  
 print("There is a significant relationship between Difference\_Ms\_P1 and Difference\_Ms\_P2.")  
else:  
 print("There is no significant relationship between Difference\_Ms\_P1 and Difference\_Ms\_P2.")

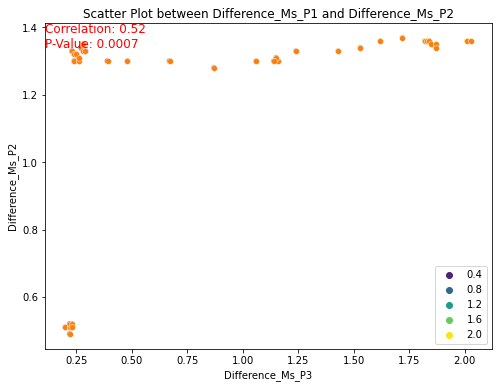
Pearson Correlation Coefficient: 0.47543154870965776  
P-Value: 0.00222261344584487



There is a significant relationship between Difference\_Ms\_P1 and Difference\_Ms\_P2.

from scipy.stats import pearsonr  
import seaborn as sns  
  
  
# Calculate Pearson correlation coefficient  
correlation\_coefficient, p\_value = pearsonr(df['Difference\_Ms\_P3'], df['Difference\_Ms\_P2'])  
  
# Display the correlation coefficient and p-value  
print(f"Pearson Correlation Coefficient: {correlation\_coefficient}")  
print(f"P-Value: {p\_value}")  
  
# Plot a scatter plot to visualize the relationship  
plt.figure(figsize=(8, 6))  
  
# Scatter plot with different colors for each variable  
sns.scatterplot(x='Difference\_Ms\_P3', y='Difference\_Ms\_P2', data=df, hue=df['Difference\_Ms\_P3'], palette='viridis')  
  
# Annotate the plot with correlation coefficient and p-value  
plt.text(plt.xlim()[0], plt.ylim()[1], f'Correlation: {correlation\_coefficient:.2f}\nP-Value: {p\_value:.4f}',  
 horizontalalignment='left', verticalalignment='top', fontsize=12, color='red')  
  
sns.scatterplot(x='Difference\_Ms\_P3', y='Difference\_Ms\_P2', data=df)  
plt.title('Scatter Plot between Difference\_Ms\_P1 and Difference\_Ms\_P2')  
plt.show()  
  
# Check significance based on the p-value  
alpha = 0.05 # significance level  
if p\_value < alpha:  
 print("There is a significant relationship between Difference\_Ms\_P3 and Difference\_Ms\_P2.")  
else:  
 print("There is no significant relationship between Difference\_Ms\_P3 and Difference\_Ms\_P2.")

Pearson Correlation Coefficient: 0.5181271868993695  
P-Value: 0.0007285554368866722



There is a significant relationship between Difference\_Ms\_P3 and Difference\_Ms\_P2.

pip install sweetviz

Note: you may need to restart the kernel to use updated packages.

WARNING: Retrying (Retry(total=4, connect=None, read=None, redirect=None, status=None)) after connection broken by 'NewConnectionError('<pip.\_vendor.urllib3.connection.HTTPSConnection object at 0x00000234FCA561F0>: Failed to establish a new connection: [WinError 10051] A socket operation was attempted to an unreachable network')': /simple/sweetviz/  
WARNING: Retrying (Retry(total=3, connect=None, read=None, redirect=None, status=None)) after connection broken by 'NewConnectionError('<pip.\_vendor.urllib3.connection.HTTPSConnection object at 0x00000234FCA56730>: Failed to establish a new connection: [Errno 11001] getaddrinfo failed')': /simple/sweetviz/  
WARNING: Retrying (Retry(total=2, connect=None, read=None, redirect=None, status=None)) after connection broken by 'NewConnectionError('<pip.\_vendor.urllib3.connection.HTTPSConnection object at 0x00000234FCA56940>: Failed to establish a new connection: [Errno 11001] getaddrinfo failed')': /simple/sweetviz/  
WARNING: Retrying (Retry(total=1, connect=None, read=None, redirect=None, status=None)) after connection broken by 'NewConnectionError('<pip.\_vendor.urllib3.connection.HTTPSConnection object at 0x00000234FCA56AF0>: Failed to establish a new connection: [Errno 11001] getaddrinfo failed')': /simple/sweetviz/  
WARNING: Retrying (Retry(total=0, connect=None, read=None, redirect=None, status=None)) after connection broken by 'NewConnectionError('<pip.\_vendor.urllib3.connection.HTTPSConnection object at 0x00000234FCA56CA0>: Failed to establish a new connection: [Errno 11001] getaddrinfo failed')': /simple/sweetviz/  
ERROR: Could not find a version that satisfies the requirement sweetviz (from versions: none)  
ERROR: No matching distribution found for sweetviz