Advance Data Science - 1

Sia Vashist

20190802107

Dataset : Electronics Sales

Aim: To perform hypothesis testing on case studies

- ### Case 1:

In [15]:
```
import pandas as pd
from scipy.stats import ttest_ind
```

In [16]:
```
# Load the dataset
data = pd.read_csv('merged_sales_data.csv')
display(data)
```

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address
0	295665	Macbook Pro Laptop	1	1700	12/30/19 00:01	136 Church St, New York City, NY 10001
1	295666	LG Washing Machine	1	600.0	12/29/19 07:03	562 2nd St, New York City, NY 10001
2	295667	USB-C Charging Cable	1	11.95	12/12/19 18:21	277 Main St, New York City, NY 10001
3	295668	27in FHD Monitor	1	149.99	12/22/19 15:13	410 6th St, San Francisco, CA 94016
4	295669	USB-C Charging Cable	1	11.95	12/18/19 12:38	43 Hill St, Atlanta, GA 30301
...	...	...	...	...	...	...
186845	222905	AAA Batteries (4-pack)	1	2.99	06/07/19 19:02	795 Pine St, Boston, MA 02215
186846	222906	27in FHD Monitor	1	149.99	06/01/19 19:29	495 North St, New York City, NY 10001
186847	222907	USB-C Charging Cable	1	11.95	06/22/19 18:57	319 Ridge St, San Francisco, CA 94016
186848	222908	USB-C Charging Cable	1	11.95	06/26/19 18:35	916 Main St, San Francisco, CA 94016
186849	222909	AAA Batteries (4-pack)	1	2.99	06/25/19 14:33	209 11th St, Atlanta, GA 30301

186850 rows × 6 columns

In [17]:
```
# Checking for missing values
missing_values = data.isnull().sum()
missing_values
```

Out[17]:
```
Order ID            545
Product             545
Quantity Ordered    545
Price Each          545
Order Date          545
Purchase Address    545
dtype: int64
```

In [18]:
```
# Handling missing values
# Removing rows with missing values
data_cleaned = data.dropna()
```

In [19]:
```
# Removing rows with invalid 'Order Date' values
data = data[data['Order Date'] != 'Order Date']

# Converting 'ORDER DATE' to datetime with format specification
data['Order Date'] = pd.to_datetime(data['Order Date'], format='%m/%d/%y %H:%M')
```

In [20]:
```
#first few rows of the cleaned dataset
```

```
display(data_cleaned)
```

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address
0	295665	Macbook Pro Laptop	1	1700	12/30/19 00:01	136 Church St, New York City, NY 10001
1	295666	LG Washing Machine	1	600.0	12/29/19 07:03	562 2nd St, New York City, NY 10001
2	295667	USB-C Charging Cable	1	11.95	12/12/19 18:21	277 Main St, New York City, NY 10001
3	295668	27in FHD Monitor	1	149.99	12/22/19 15:13	410 6th St, San Francisco, CA 94016
4	295669	USB-C Charging Cable	1	11.95	12/18/19 12:38	43 Hill St, Atlanta, GA 30301
...	...	...	...	...	...	...
186845	222905	AAA Batteries (4-pack)	1	2.99	06/07/19 19:02	795 Pine St, Boston, MA 02215
186846	222906	27in FHD Monitor	1	149.99	06/01/19 19:29	495 North St, New York City, NY 10001
186847	222907	USB-C Charging Cable	1	11.95	06/22/19 18:57	319 Ridge St, San Francisco, CA 94016
186848	222908	USB-C Charging Cable	1	11.95	06/26/19 18:35	916 Main St, San Francisco, CA 94016
186849	222909	AAA Batteries (4-pack)	1	2.99	06/25/19 14:33	209 11th St, Atlanta, GA 30301

186305 rows × 6 columns

In [21]:

```
# Converting 'Price Each' and 'Quantity Ordered' to numeric
data['Price Each'] = pd.to_numeric(data['Price Each'], errors='coerce')
data['Quantity Ordered'] = pd.to_numeric(data['Quantity Ordered'], errors='coerce')
```

The errors='coerce' parameter will convert any non-numeric values to NaN

In [22]:

```
# Droping rows with NaN values in 'Price Each' or 'Quantity Ordered'
data.dropna(subset=['Price Each', 'Quantity Ordered'], inplace=True)
```

In [23]:

```
# Calculating historical average sales
historical_avg_sales = (data['Price Each'] * data['Quantity Ordered']).mean()

# Filtering data for the new campaign period
new_campaign_data = data[data['Order Date'] >= '2020-01-01']

# Average sales during the new campaign
new_campaign_avg_sales = (new_campaign_data['Price Each'] * new_campaign_data['Quantit
```

In [24]:

```
# Descriptive Statistics
descriptive_stats = data[['Price Each', 'Quantity Ordered']].describe()
descriptive_stats
```

Out[24]:

	Price Each	Quantity Ordered
count	185950.000000	185950.000000
mean	184.399735	1.124383
std	332.731330	0.442793
min	2.990000	1.000000
25%	11.950000	1.000000
50%	14.950000	1.000000
75%	150.000000	1.000000
max	1700.000000	9.000000

In [25]:

```
print("Historical Average Sales:", historical_avg_sales)
print("New Campaign Average Sales:", new_campaign_avg_sales)
```

```
Historical Average Sales: 185.490916751815
New Campaign Average Sales: 255.00852941176473
```

In [26]:

```
# t-test for independent samples
t_statistic, p_value = ttest_ind(data['Price Each'] * data['Quantity Ordered'], new_ca
print("\nT-statistic:", t_statistic)
print("P-value:", p_value)
```

```
T-statistic: -1.0770250277117837
P-value: 0.28927761858512807
```

In [29]:

```
# Set significance level
alpha = 0.05

# Compare p-value with significance level
if p_value < alpha:
    print("Reject the null hypothesis: The new marketing campaign does not lead to an
else:
    print("Fail to reject the null hypothesis: There is statistically significant incr
```

```
Fail to reject the null hypothesis: There is statistically significant increase in sa
les due to the new marketing campaign.
```

In [1]:
```
import pandas as pd

# Loading the dataset
dataset2 = pd.read_csv('data_linear.csv')
display(dataset2)
```

	Temperature Test	Quality Test	Result
0	34.623660	78.024693	0
1	30.286711	43.894998	0
2	35.847409	72.902198	0
3	60.182599	86.308552	1
4	79.032736	75.344376	1
...	...	...	...
95	83.489163	48.380286	1
96	42.261701	87.103851	1
97	99.315009	68.775409	1
98	55.340018	64.931938	1
99	74.775893	89.529813	1

100 rows × 3 columns

In [2]:
```
# Descriptive statistics for Temperature Test and Quality Test
descriptive_stats = dataset2[['Temperature Test', 'Quality Test']].describe()
print(descriptive_stats)
```

```
       Temperature Test  Quality Test
count        100.000000    100.000000
mean          65.644274     66.221998
std           19.458222     18.582783
min           30.058822     30.603263
25%           50.919511     48.179205
50%           67.032988     67.682381
75%           80.212529     79.360605
max           99.827858     98.869436
```

In [3]:
```
# Handling Missing Values
# Checking for missing values
missing_values = dataset2.isnull().sum()
data_cleaned1 = dataset2.dropna()

print("Missing Values:\n", missing_values)
```

```
Missing Values:
 Temperature Test    0
Quality Test        0
Result              0
dtype: int64
```

In [6]:
```
from scipy.stats import ttest_ind

# Separate data for the current and previous manufacturing processes
current_quality = data_cleaned1[data_cleaned1['Result'] == 0]['Quality Test']
previous_quality = data_cleaned1[data_cleaned1['Result'] == 1]['Quality Test']

# Perform t-test for independent samples
t_statistic, p_value = ttest_ind(current_quality, previous_quality)

# Significance level
alpha = 0.05

# Interpretation
if p_value < alpha:
    hypothesis_result = "Reject the null hypothesis: Reverting to the previous manufac
else:
    hypothesis_result = "Fail to reject the null hypothesis: There is no statistically

# Print results
print("T-statistic:", t_statistic)
print("P-value:", p_value)
print("Hypothesis Result:", hypothesis_result)
```

```
T-statistic: -5.905665563839061
P-value: 5.0730596140718295e-08
Hypothesis Result: Reject the null hypothesis: Reverting to the previous manufacturin
g process improves product quality.
```

Conclusion:

As a result, we have successfully completed hypothesis testing on two case studies involving electronics companies.