Lab Assignment 10

Team Members: JENIL PATEL, Mithlesh Singla
Team No.: 36

Github Link: LAB-10

PART - 1

```
Missing values before cleaning:
id                    0
full_name             0
age                4766
gender             4693
device_type        2000
ad_position        2000
browsing_history   4782
time_of_day        2000
click                 0
dtype: int64
```

```
    # Handle missing values (fill or drop depending on extent)
    if df.isnull().sum().sum() > 0:
        # Fill categorical columns with mode
        for col in ['gender', 'device_type', 'ad_position', 'browsing_history', 'time_of_day']:
            if df[col].isnull().sum() > 0:
                df[col] = df[col].fillna(df[col].mode()[0])

        # Fill numerical columns with median
        for col in ['age']:
            if df[col].isnull().sum() > 0:
                df[col] = df[col].fillna(df[col].median())

        print("\nMissing values after cleaning:")
        print(df.isnull().sum())
```

```
Missing values after cleaning:
id                 0
full_name          0
age                0
gender             0
```

1b. Convert categorical columns (e.g., gender, ad_position)

```python
# One-hot encode other categorical variables if needed for future modeling
categorical_cols = ['gender', 'device_type', 'browsing_history', 'time_of_day']
df_encoded = pd.get_dummies(df, columns=categorical_cols, drop_first=True)

print(f"After preprocessing, dataset has {df_encoded.shape[1]} columns")
```

Python

```
After preprocessing, dataset has 17 columns
```

```python
df_encoded.head()
```

Python

	id	full_name	age	ad_position	click	ad_position_numeric	gender_Male	gender_Non-Binary	device_type_Mobile	device_type_Tablet	browsing_history_Entertainmen
0	670	User670	22.0	Top	1	0	False	False	False	False	Fals
1	3044	User3044	39.5	Top	1	0	True	False	False	False	Tru
2	5912	User5912	41.0	Side	1	2	False	True	False	False	Fals
3	5418	User5418	34.0	Bottom	1	1	True	False	False	False	Tru
4	9452	User9452	39.0	Bottom	0	1	False	True	False	False	Fals

a. Group A: Users with ad_position = 0 (Top)

b. Group B: Users with ad_position = 1 (Bottom)

```python
# Filter for Top and Bottom positions only (excluding Side if present)
ab_test_df = df[df['ad_position'].isin(['Top', 'Bottom'])]

group_a = ab_test_df[ab_test_df['ad_position'] == 'Top']
group_b = ab_test_df[ab_test_df['ad_position'] == 'Bottom']

print(f"Group A (Top position) size: {len(group_a)}")
print(f"Group B (Bottom position) size: {len(group_b)}")
```

```
Group A (Top position) size: 2597
Group B (Bottom position) size: 4817
```

```python
# Calculate success counts (clicks) for each group
success_a = group_a['click'].sum()
success_b = group_b['click'].sum()

# Total observations in each group
nobs_a = len(group_a)
nobs_b = len(group_b)

# Print the click-through rates
ctr_a = success_a / nobs_a * 100
ctr_b = success_b / nobs_b * 100
print(f"Click-through rate for Group A (Top): {ctr_a:.2f}%")
print(f"Click-through rate for Group B (Bottom): {ctr_b:.2f}%")
```

```
Click-through rate for Group A (Top): 63.50%
Click-through rate for Group B (Bottom): 66.81%
```

5. Print the following:

The z-score [10 points]

The p-value [10 points]

```
from statsmodels.stats.proportion import proportions_ztest

# Perform z-test
count = np.array([success_a, success_b])
nobs = np.array([nobs_a, nobs_b])

z_stat, p_val = proportions_ztest(count, nobs)

print(f"\nZ-score: {z_stat:.4f}")
print(f"P-value: {p_val:.4f}")
```

```
Z-score: -2.8620
P-value: 0.0042
```

```
Interpretation:
The p-value (0.0042) is less than the significance level (0.05).
Therefore, we reject the null hypothesis.
CONCLUSION: There is a statistically significant difference in click-through rates between ads positioned
at the top versus the bottom of the webpage.
Bottom positioned ads have a 3.31% higher click-through rate than top positioned ads.
```

PART - 2

```
train_df = pd.read_csv('train.csv')
test1_df = pd.read_csv('test1.csv')
test2_df = pd.read_csv('test2.csv')

print(f"Train dataset shape: {train_df.shape}")
print(f"Test1 dataset shape: {test1_df.shape}")
print(f"Test2 dataset shape: {test2_df.shape}")
```

```
Train dataset shape: (3200, 18)
Test1 dataset shape: (800, 18)
Test2 dataset shape: (800, 18)
```

```
train_df.head()
```

	Unnamed: 0	Date	Time	CO(GT)	PT08.S1(CO)	NMHC(GT)	C6H6(GT)	P
0	1849	26/05/2004	19.00.00	-200	1130.0	-200.0	22,7	
1	2533	24/06/2004	07.00.00	1,2	1030.0	-200.0	6,9	
2	3047	15/07/2004	17.00.00	3,2	1164.0	-200.0	20,3	
3	805	13/04/2004	07.00.00	3,9	1496.0	524.0	19,1	
4	2962	12/07/2004	04.00.00	-200	780.0	-200.0	1,8	

```
KS Test Results for NO2(GT):
Test1 vs Train: KS statistic = 0.0191, p-value = 0.9722
Test2 vs Train: KS statistic = 0.4075, p-value = 0.0000
```

```python
alpha = 0.05  # significance level
if p_value_test1 < alpha:
    print("Test1 vs Train: Distributions are significantly different.")
else:
    print("Test1 vs Train: No significant difference in distributions.")

if p_value_test2 < alpha:
    print("Test2 vs Train: Distributions are significantly different.")
else:
    print("Test2 vs Train: No significant difference in distributions.")
```

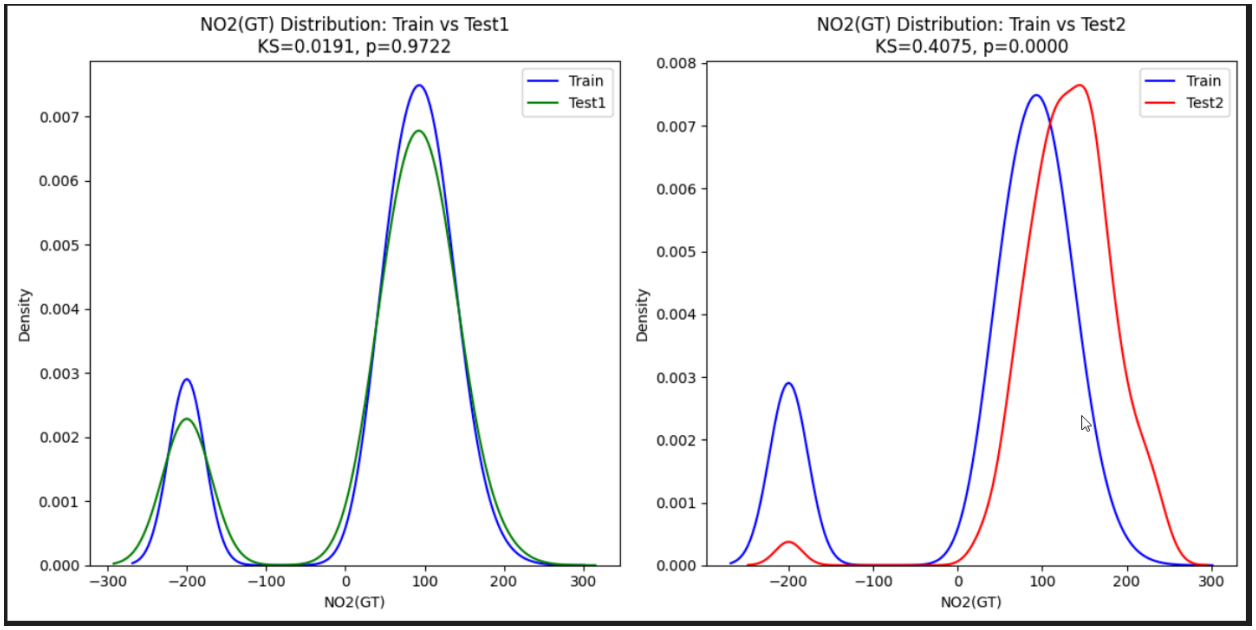
[2] ✓ 0.0s

```
Test1 vs Train: No significant difference in distributions.
Test2 vs Train: Distributions are significantly different.
```

```python
# Create a DataFrame to display the results
results_df = pd.DataFrame({
    'Feature': columns,
    'KS_Stat_Test1': ks_test1,
    'p_value_Test1': p_test1,
    'KS_Stat_Test2': ks_test2,
    'p_value_Test2': p_test2
})

print(results_df.to_string(index=False, float_format=lambda x: f"{x:.4f}"))
```

Feature	KS_Stat_Test1	p_value_Test1	KS_Stat_Test2	p_value_Test2
Unnamed: 0	0.0462	0.1270	1.0000	0.0000
CO(GT)	0.0256	0.7888	0.2128	0.0000
PT08.S1(CO)	0.0328	0.4900	0.1275	0.0000
NMHC(GT)	0.0128	0.9999	0.2272	0.0000
C6H6(GT)	0.0353	0.3966	0.0556	0.0372
PT08.S2(NMHC)	0.0216	0.9235	0.1419	0.0000
NOx(GT)	0.0175	0.9885	0.5241	0.0000
PT08.S3(NOx)	0.0344	0.4304	0.3228	0.0000
NO2(GT)	0.0191	0.9722	0.4075	0.0000
PT08.S4(NO2)	0.0200	0.9574	0.5972	0.0000
PT08.S5(O3)	0.0281	0.6856	0.1366	0.0000
T	0.0184	0.9799	0.2537	0.0000
RH	0.0163	0.9953	0.1787	0.0000
AH	0.0256	0.7888	0.4019	0.0000




```python
# Conclusion
print("\nConclusion:")
if avg_ks_test1 > avg_ks_test2 and sig_diff_test1 > sig_diff_test2:
    print("Test1 exhibits stronger covariate shift relative to the training dataset.")
elif avg_ks_test2 > avg_ks_test1 and sig_diff_test2 > sig_diff_test1:
    print("Test2 exhibits stronger covariate shift relative to the training dataset.")
else:
    # If the results are mixed, look at NO2(GT) specifically as mentioned in the task
    if ks_stat_test1 > ks_stat_test2 and p_value_test1 < p_value_test2:
        print("Based on the NO2(GT) column, Test1 exhibits stronger covariate shift relative to the training dataset.")
    elif ks_stat_test2 > ks_stat_test1 and p_value_test2 < p_value_test1:
        print("Based on the NO2(GT) column, Test2 exhibits stronger covariate shift relative to the training dataset.")
    else:
        print("Results are mixed across features. Further analysis is recommended.")
```

```
Conclusion:
Test2 exhibits stronger covariate shift relative to the training dataset.
```