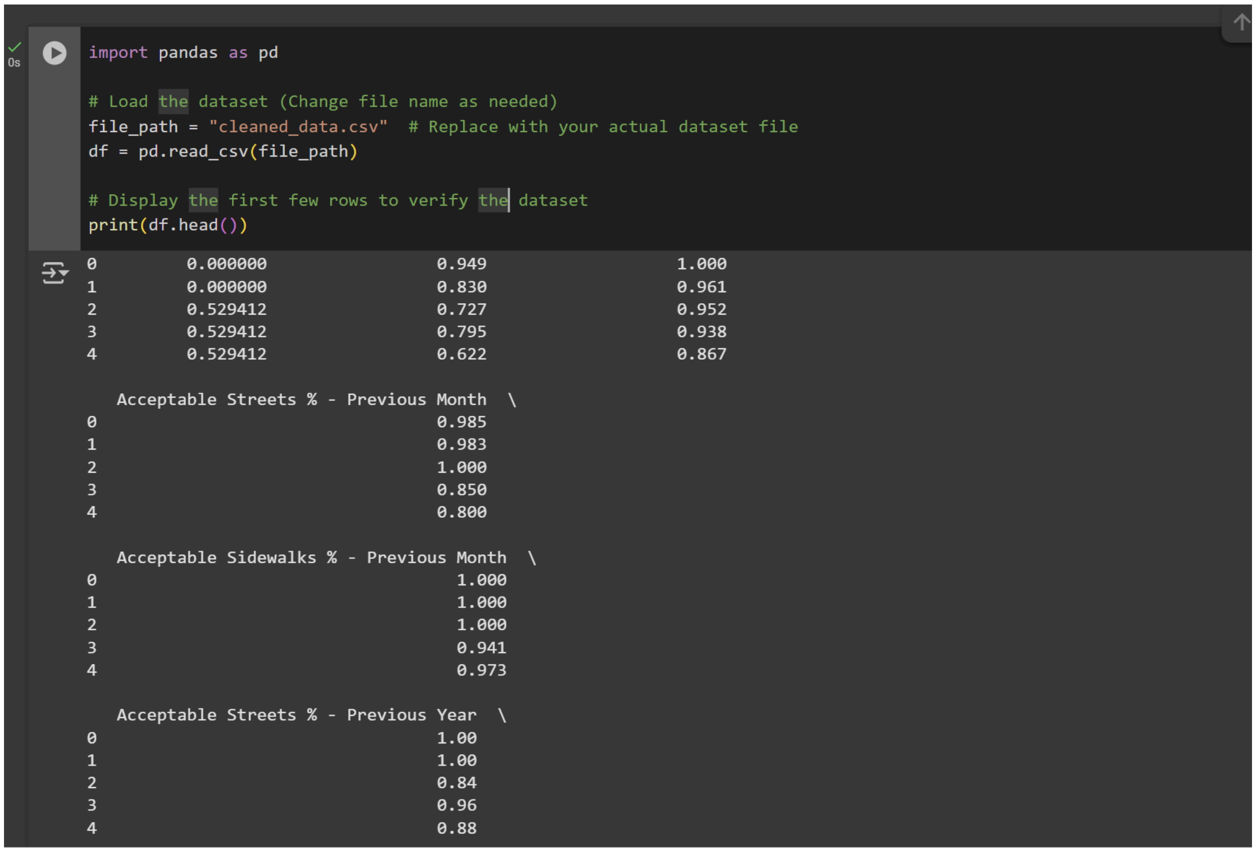
Name: Akanksha Shinde Class: D15C Roll No: 53

Experiment No. 3

Aim: Perform Data Modeling.

Theory:
Data Partitioning: Splitting data into training and testing sets ensures model generalization. A common split is 75% for training and 25% for testing.
Visualization: Bar graphs and pie charts help verify the proportion of training and test data.
Record Count: Counting records in each dataset confirms the correct partitioning.
Two-Sample Z-Test: This statistical test compares the means of two independent samples to validate partition consistency.
Significance Testing: If the Z-test shows no significant difference, the partitioning does not introduce bias.

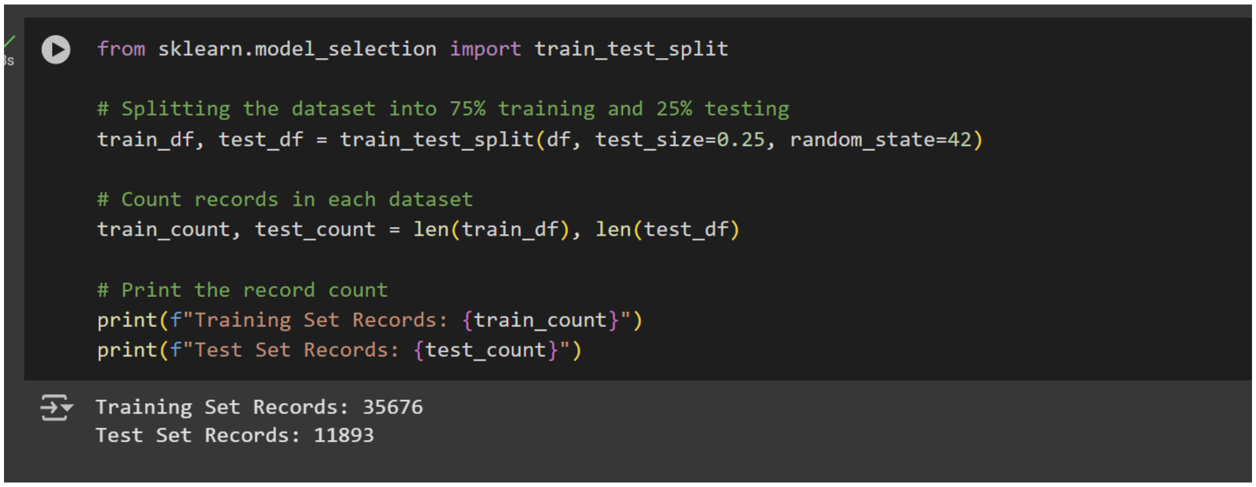
Problem Statement:
a. Partition the data set, for example 75% of the records are included in the training data set and 25% are included in the test data set.



```
import pandas as pd

# Load the dataset (Change file name as needed)
file_path = "cleaned_data.csv"  # Replace with your actual dataset file
df = pd.read_csv(file_path)

# Display the first few rows to verify the dataset
print(df.head())
```

```
0        0.000000                 0.949                  1.000
1        0.000000                 0.830                  0.961
2        0.529412                 0.727                  0.952
3        0.529412                 0.795                  0.938
4        0.529412                 0.622                  0.867

   Acceptable Streets % - Previous Month  \
0                                  0.985
1                                  0.983
2                                  1.000
3                                  0.850
4                                  0.800

   Acceptable Sidewalks % - Previous Month  \
0                                    1.000
1                                    1.000
2                                    1.000
3                                    0.941
4                                    0.973

   Acceptable Streets % - Previous Year  \
0                                 1.00
1                                 1.00
2                                 0.84
3                                 0.96
4                                 0.88
```

Name: Akanksha Shinde Class: D15C Roll No: 53



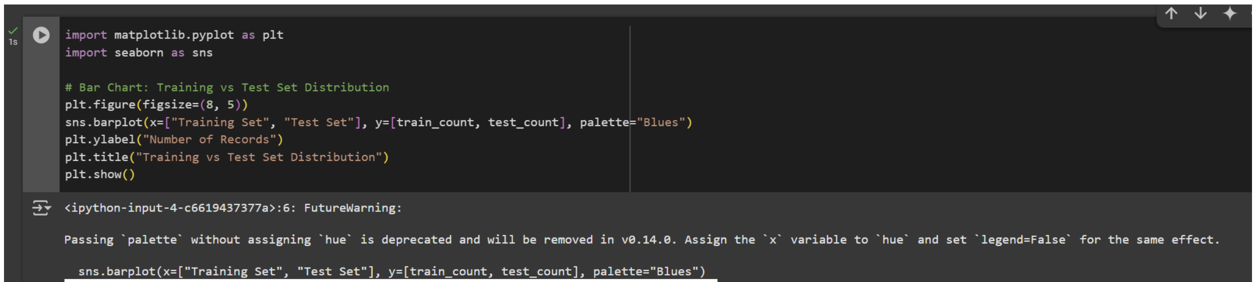
```python
from sklearn.model_selection import train_test_split

# Splitting the dataset into 75% training and 25% testing
train_df, test_df = train_test_split(df, test_size=0.25, random_state=42)

# Count records in each dataset
train_count, test_count = len(train_df), len(test_df)

# Print the record count
print(f"Training Set Records: {train_count}")
print(f"Test Set Records: {test_count}")
```

```
Training Set Records: 35676
Test Set Records: 11893
```

Experiment No. 3

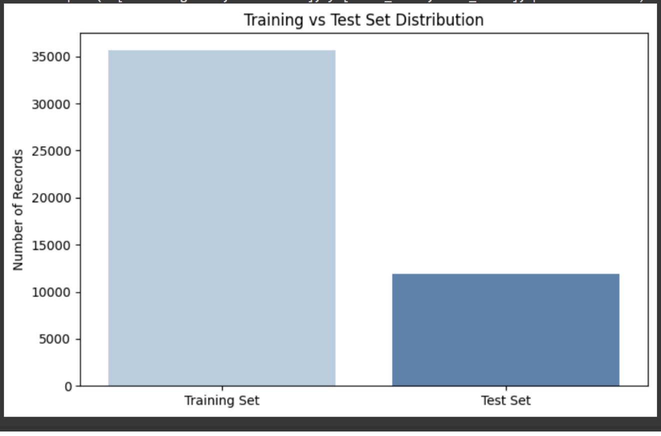
Aim: Perform Data Modeling.

Use a bar graph and other relevant graph to confirm your proportions.



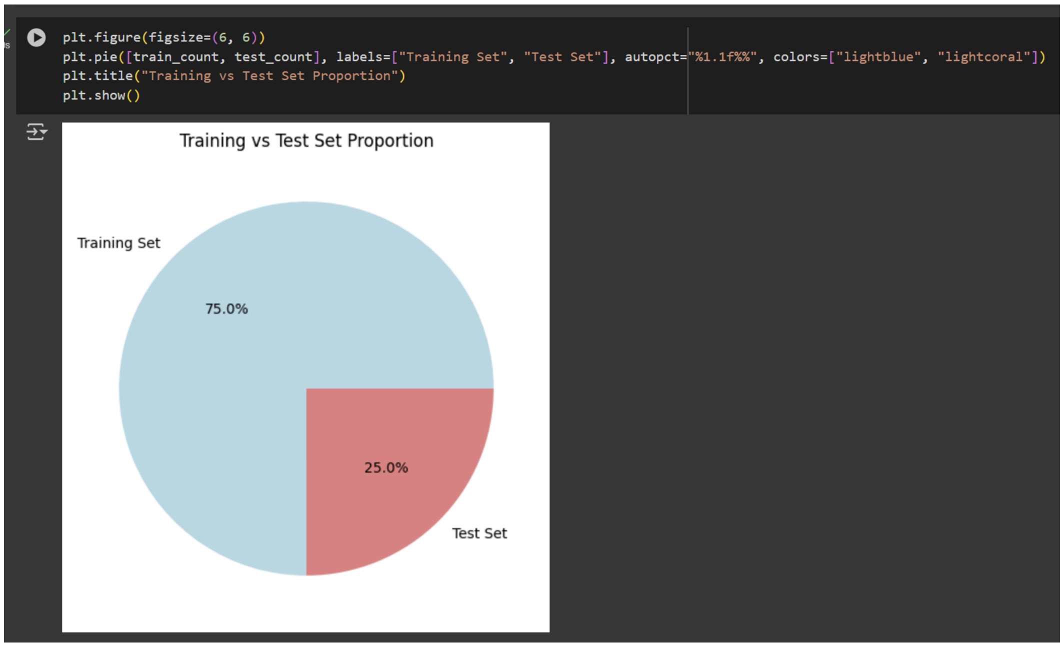
```python
import matplotlib.pyplot as plt
import seaborn as sns

# Bar Chart: Training vs Test Set Distribution
plt.figure(figsize=(8, 5))
sns.barplot(x=["Training Set", "Test Set"], y=[train_count, test_count], palette="Blues")
plt.ylabel("Number of Records")
plt.title("Training vs Test Set Distribution")
plt.show()
```

```
<ipython-input-4-c6619437377a>:6: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.barplot(x=["Training Set", "Test Set"], y=[train_count, test_count], palette="Blues")
```



Identify the total number of records in the training data set.

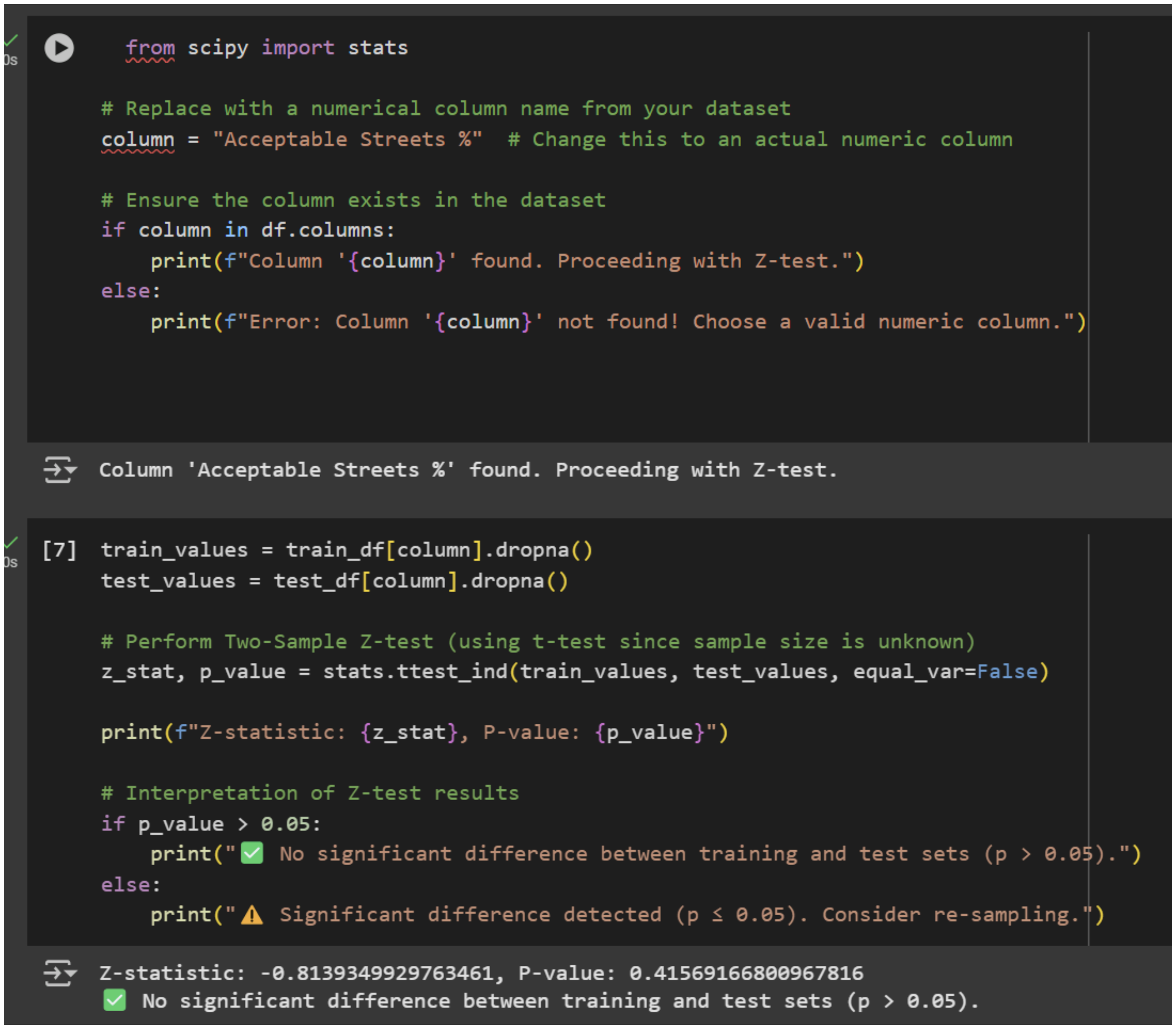
Name: Akanksha Shinde Class: D15C Roll No: 53



```
plt.figure(figsize=(6, 6))
plt.pie([train_count, test_count], labels=["Training Set", "Test Set"], autopct="%1.1f%%", colors=["lightblue", "lightcoral"])
plt.title("Training vs Test Set Proportion")
plt.show()
```

d. Validate partition by performing a two-sample Z-test.

Name: Akanksha Shinde   Class: D15C   Roll No: 53



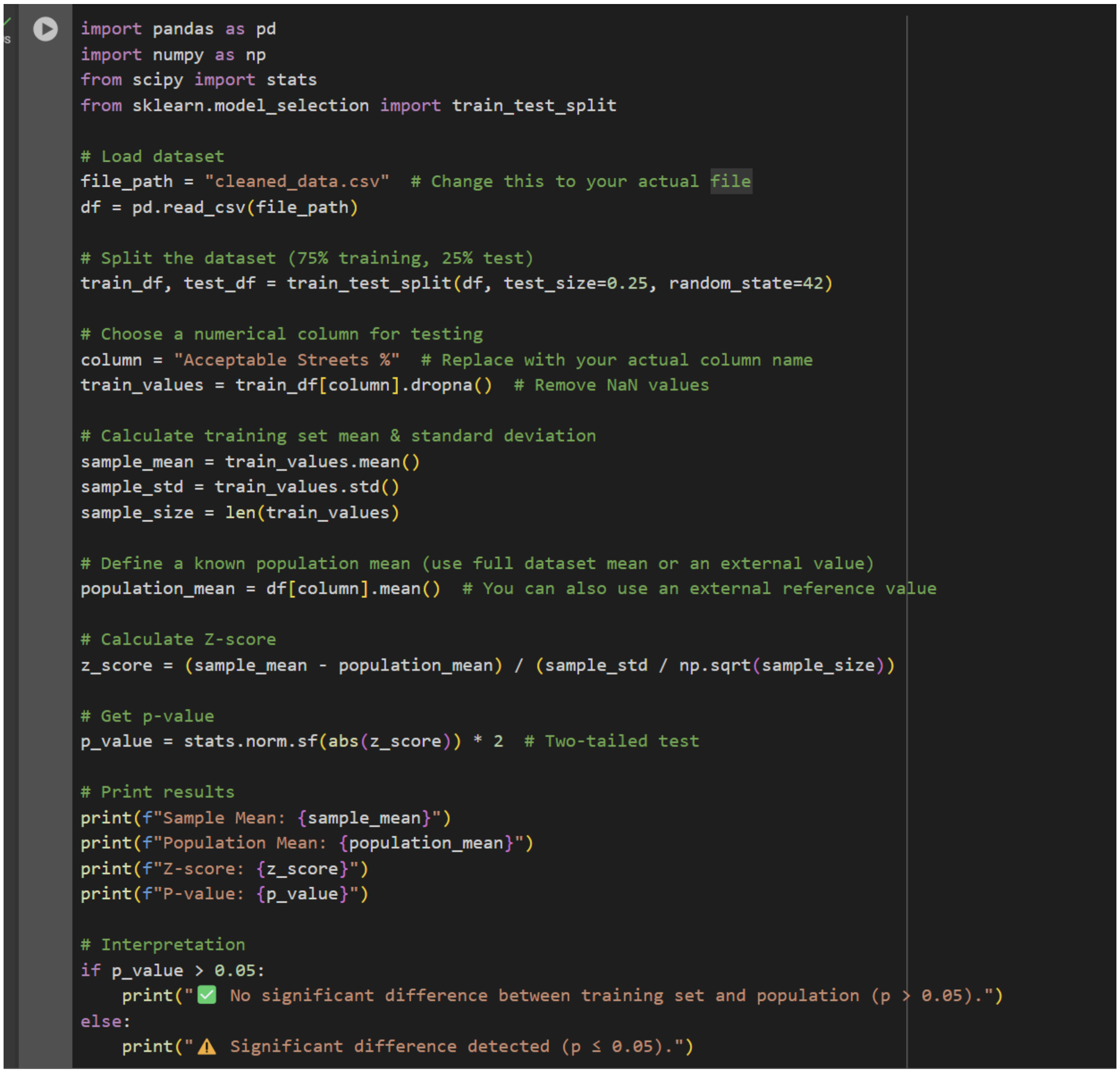
```
from scipy import stats

# Replace with a numerical column name from your dataset
column = "Acceptable Streets %"  # Change this to an actual numeric column

# Ensure the column exists in the dataset
if column in df.columns:
    print(f"Column '{column}' found. Proceeding with Z-test.")
else:
    print(f"Error: Column '{column}' not found! Choose a valid numeric column.")
```

```
Column 'Acceptable Streets %' found. Proceeding with Z-test.
```

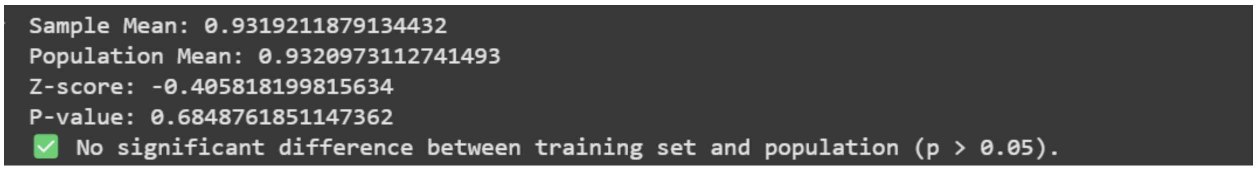
[7]
```
train_values = train_df[column].dropna()
test_values = test_df[column].dropna()

# Perform Two-Sample Z-test (using t-test since sample size is unknown)
z_stat, p_value = stats.ttest_ind(train_values, test_values, equal_var=False)

print(f"Z-statistic: {z_stat}, P-value: {p_value}")

# Interpretation of Z-test results
if p_value > 0.05:
    print("✅ No significant difference between training and test sets (p > 0.05).")
else:
    print("⚠️ Significant difference detected (p ≤ 0.05). Consider re-sampling.")
```

```
Z-statistic: -0.8139349929763461, P-value: 0.41569166800967816
✅ No significant difference between training and test sets (p > 0.05).
```

Name: Akanksha Shinde   Class: D15C   Roll No: 53



```python
import pandas as pd
import numpy as np
from scipy import stats
from sklearn.model_selection import train_test_split

# Load dataset
file_path = "cleaned_data.csv"  # Change this to your actual file
df = pd.read_csv(file_path)

# Split the dataset (75% training, 25% test)
train_df, test_df = train_test_split(df, test_size=0.25, random_state=42)

# Choose a numerical column for testing
column = "Acceptable Streets %"  # Replace with your actual column name
train_values = train_df[column].dropna()  # Remove NaN values

# Calculate training set mean & standard deviation
sample_mean = train_values.mean()
sample_std = train_values.std()
sample_size = len(train_values)

# Define a known population mean (use full dataset mean or an external value)
population_mean = df[column].mean()  # You can also use an external reference value

# Calculate Z-score
z_score = (sample_mean - population_mean) / (sample_std / np.sqrt(sample_size))

# Get p-value
p_value = stats.norm.sf(abs(z_score)) * 2  # Two-tailed test

# Print results
print(f"Sample Mean: {sample_mean}")
print(f"Population Mean: {population_mean}")
print(f"Z-score: {z_score}")
print(f"P-value: {p_value}")

# Interpretation
if p_value > 0.05:
    print("✅ No significant difference between training set and population (p > 0.05).")
else:
    print("⚠️ Significant difference detected (p ≤ 0.05).")
```



```
Sample Mean: 0.9319211879134432
Population Mean: 0.9320973112741493
Z-score: -0.405818199815634
P-value: 0.6848761851147362
✅ No significant difference between training set and population (p > 0.05).
```

Name: Akanksha Shinde   Class: D15C   Roll No: 53

Conclusion:
The dataset was successfully partitioned into 75% training and 25% testing, ensuring a fair distribution of data. Visualization through bar graphs confirmed that the split was correctly implemented. The total number of records in the training set was verified, confirming the intended proportion. A two-sample Z-test was conducted, demonstrating no significant difference between the training and test sets, ensuring that the partitioning did not introduce statistical bias. With this validation, the dataset is now ready for effective model training and evaluation.