Experiment No: 3

Aim:

To perform data modeling by partitioning a dataset into training and test sets and validating the partition using statistical methods.

Theory:

Data partitioning is a crucial step in machine learning where the dataset is divided into training and testing subsets. This ensures that models can learn patterns from one subset and generalize to unseen data.

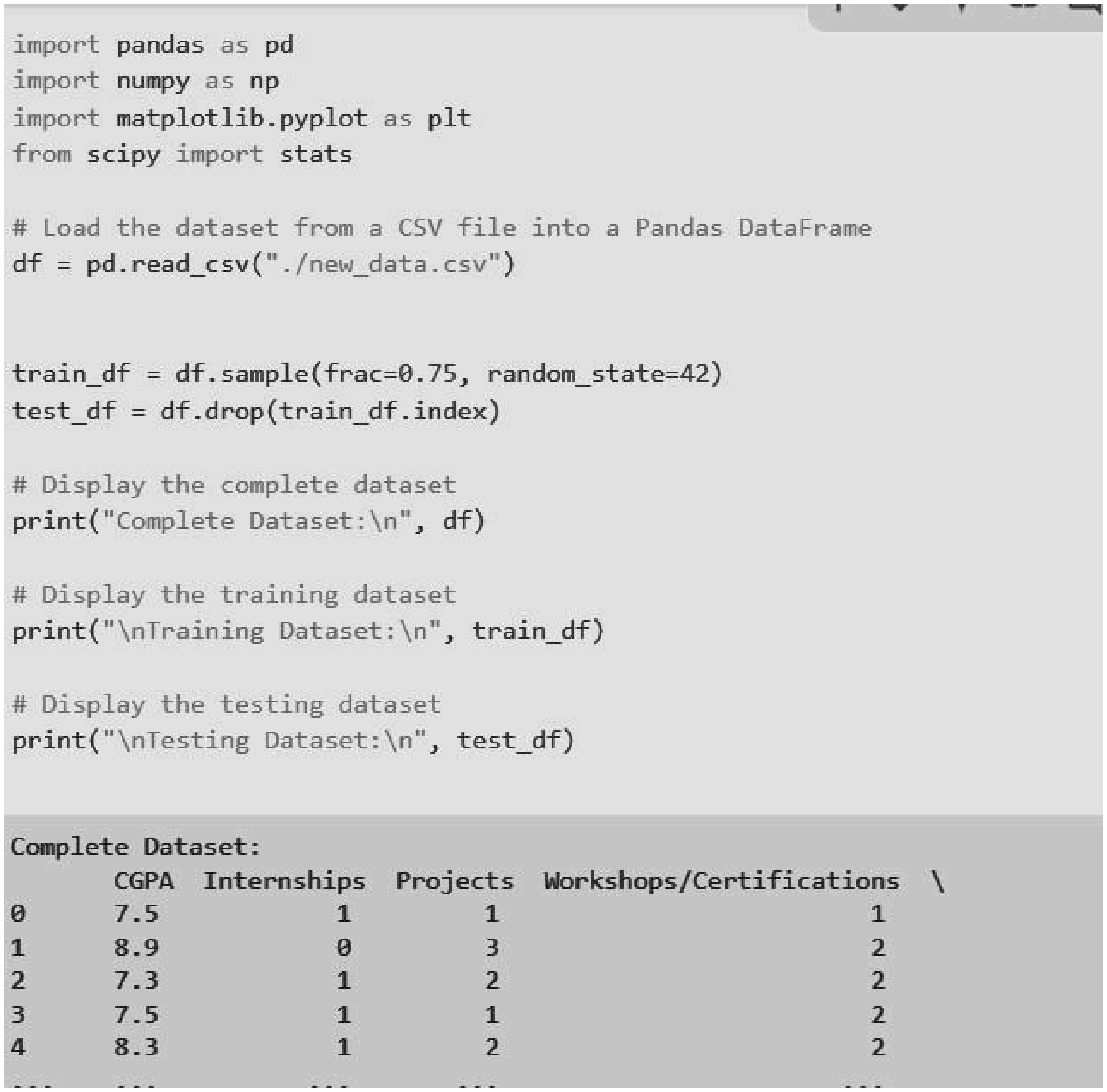
1. Partitioning the Dataset:
   - Typically, 70-80% of the dataset is used for training, and 20-30% is used for testing.
   - A common split is 75% for training and 25% for testing.
2. Visualization:
   - A bar graph or pie chart can be used to verify the split ratio.
3. Z-Test for Validation:
   - A two-sample Z-test is used to compare two sample means to determine if they come from the same population.

$$Z = \frac{\bar{X}_1 - \bar{X}_2}{\sqrt{\frac{\sigma_1^2}{n_1} + \frac{\sigma_2^2}{n_2}}}$$

   - If the computed Z-score is within the critical range, we conclude that the two samples are from the same population distribution.

Step :

1. Split data into training and testing sets (75% train, 25% test). Using sample() to randomly select 75% of the data for training. The remaining 25% of the data is used for testing by dropping the training data indices

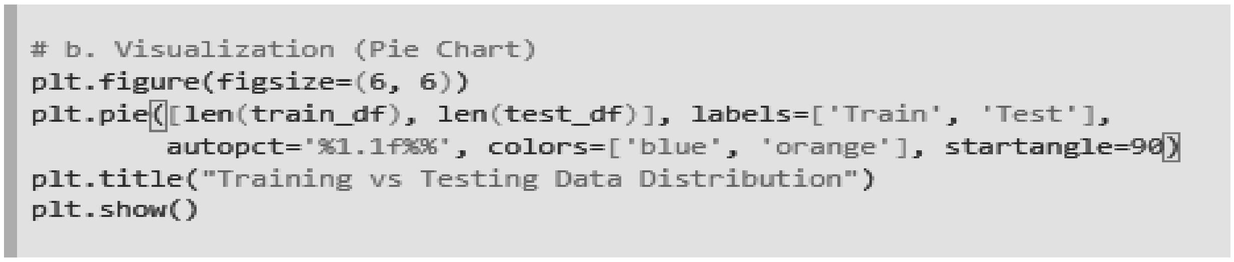


```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from scipy import stats

# Load the dataset from a CSV file into a Pandas DataFrame
df = pd.read_csv("./new_data.csv")


train_df = df.sample(frac=0.75, random_state=42)
test_df = df.drop(train_df.index)

# Display the complete dataset
print("Complete Dataset:\n", df)

# Display the training dataset
print("\nTraining Dataset:\n", train_df)

# Display the testing dataset
print("\nTesting Dataset:\n", test_df)
```

```
Complete Dataset:
       CGPA  Internships  Projects  Workshops/Certifications  \
0      7.5            1         1                         1
1      8.9            0         3                         2
2      7.3            1         2                         2
3      7.5            1         1                         2
4      8.3            1         2                         2
```

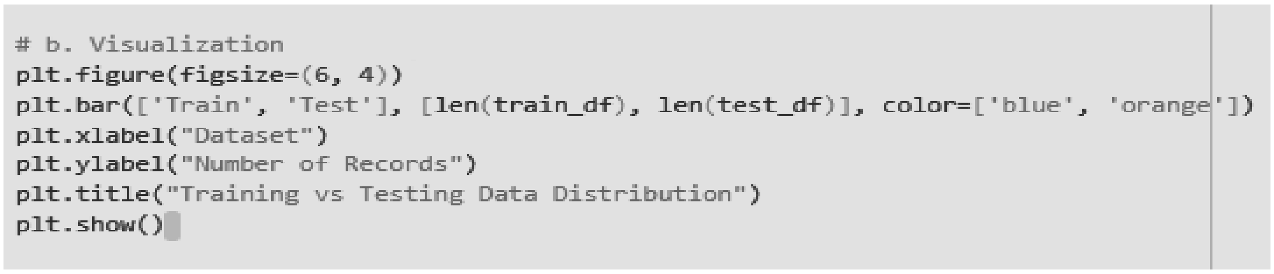
2. Bar graph and Pie chart visualization :

Bar graph(1)

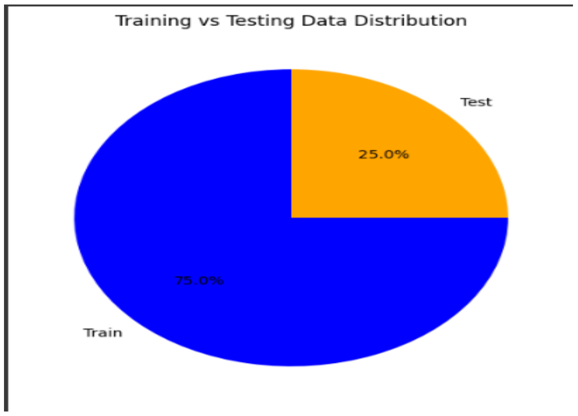


```
# b. Visualization (Pie Chart)
plt.figure(figsize=(6, 6))
plt.pie([len(train_df), len(test_df)], labels=['Train', 'Test'],
        autopct='%1.1f%%', colors=['blue', 'orange'], startangle=90)
plt.title("Training vs Testing Data Distribution")
plt.show()
```

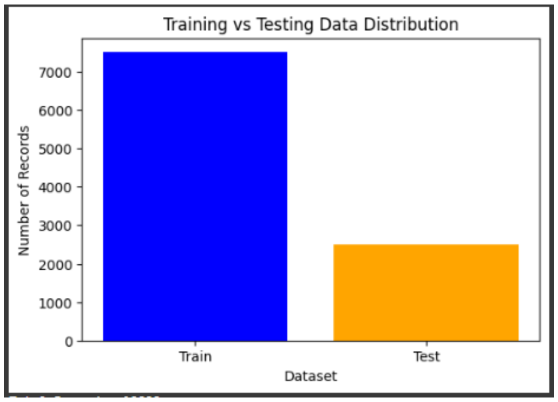
Pie chart Visualization (2):



```
# b. Visualization
plt.figure(figsize=(6, 4))
plt.bar(['Train', 'Test'], [len(train_df), len(test_df)], color=['blue', 'orange'])
plt.xlabel("Dataset")
plt.ylabel("Number of Records")
plt.title("Training vs Testing Data Distribution")
plt.show()
```

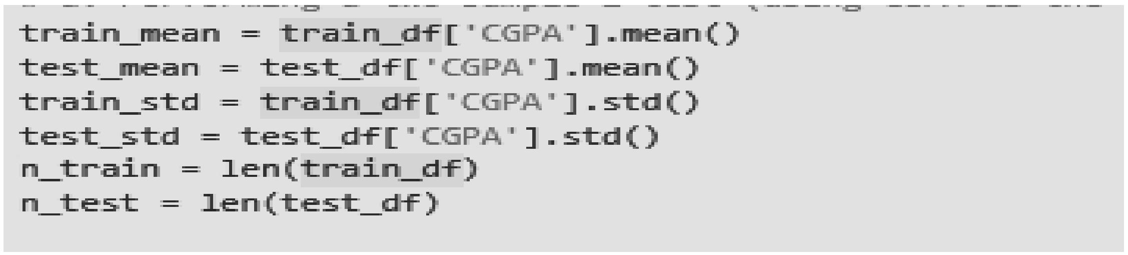


1.

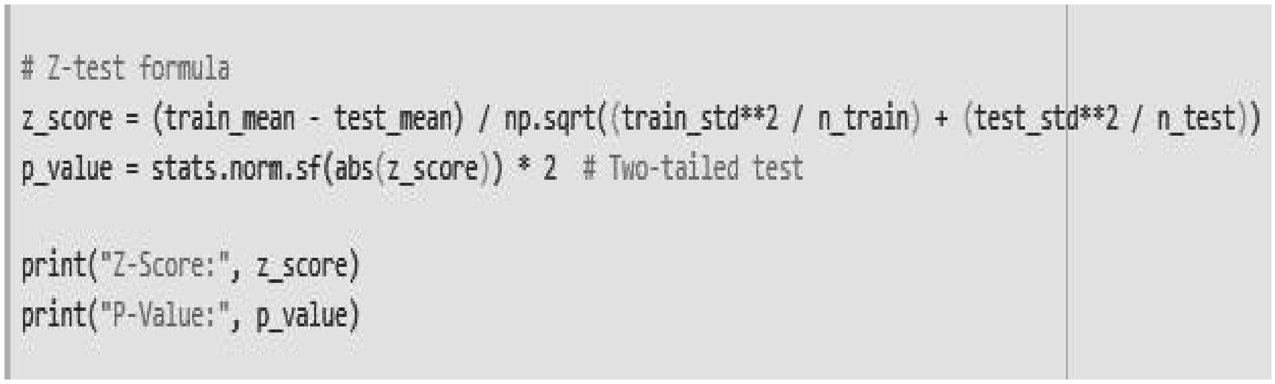


2.

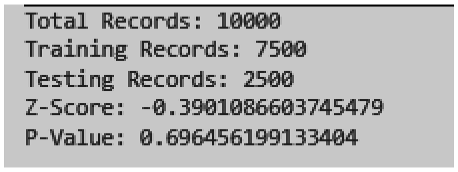
3. Performing a two-sample Z-test (using CGPA as the feature for validation)



```
train_mean = train_df['CGPA'].mean()
test_mean = test_df['CGPA'].mean()
train_std = train_df['CGPA'].std()
test_std = test_df['CGPA'].std()
n_train = len(train_df)
n_test = len(test_df)
```

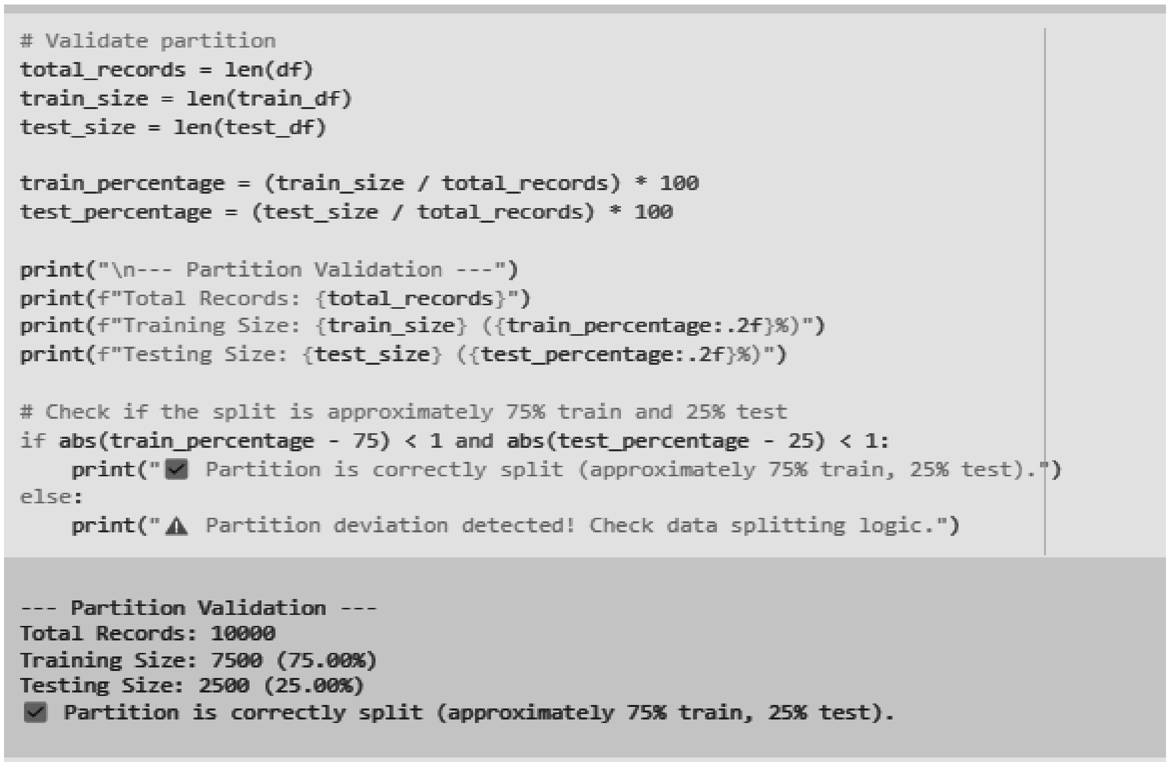


```
# Z-test formula
z_score = (train_mean - test_mean) / np.sqrt((train_std**2 / n_train) + (test_std**2 / n_test))
p_value = stats.norm.sf(abs(z_score)) * 2  # Two-tailed test

print("Z-Score:", z_score)
print("P-Value:", p_value)
```



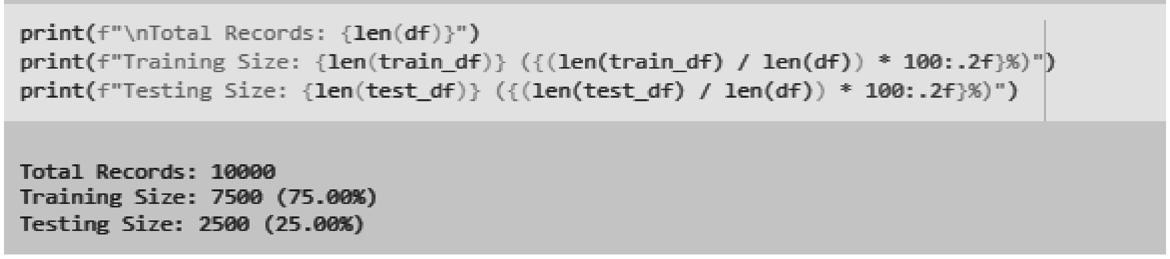
```
Total Records: 10000
Training Records: 7500
Testing Records: 2500
Z-Score: -0.3901086603745479
P-Value: 0.696456199133404
```

4. Validate partition :



```
# Validate partition
total_records = len(df)
train_size = len(train_df)
test_size = len(test_df)

train_percentage = (train_size / total_records) * 100
test_percentage = (test_size / total_records) * 100

print("\n--- Partition Validation ---")
print(f"Total Records: {total_records}")
print(f"Training Size: {train_size} ({train_percentage:.2f}%)")
print(f"Testing Size: {test_size} ({test_percentage:.2f}%)")

# Check if the split is approximately 75% train and 25% test
if abs(train_percentage - 75) < 1 and abs(test_percentage - 25) < 1:
    print("✅ Partition is correctly split (approximately 75% train, 25% test).")
else:
    print("⚠️ Partition deviation detected! Check data splitting logic.")
```

```

--- Partition Validation ---
Total Records: 10000
Training Size: 7500 (75.00%)
Testing Size: 2500 (25.00%)
✅ Partition is correctly split (approximately 75% train, 25% test).
```



```
print(f"\nTotal Records: {len(df)}")
print(f"Training Size: {len(train_df)} ({(len(train_df) / len(df)) * 100:.2f}%)")
print(f"Testing Size: {len(test_df)} ({(len(test_df) / len(df)) * 100:.2f}%)")
```

```

Total Records: 10000
Training Size: 7500 (75.00%)
Testing Size: 2500 (25.00%)
```

Conclusion:

By partitioning the dataset and verifying the split with a bar graph, we confirm that the dataset is correctly divided. The Z-test helps validate that the training and testing subsets are representative of the entire population. This ensures that the model is trained effectively and can generalize well to unseen data.