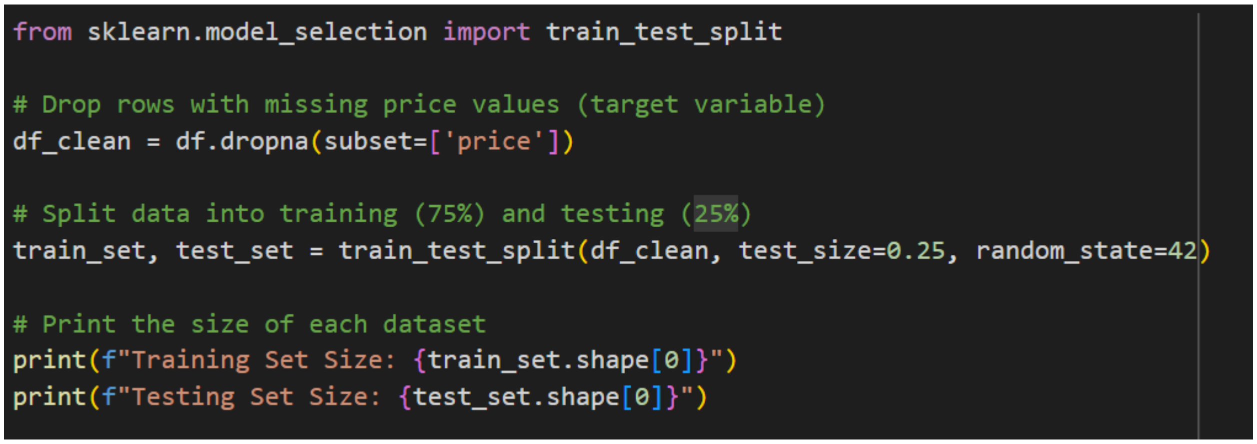
Name-Laukik Padgaonkar **D15C** **Roll no-37**

Aim: Perform Data Modeling.
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.
b. Use a bar graph and other relevant graphs to confirm your proportions.
c. Identify the total number of records in the training data set.
d. Validate partition by performing a two-sample Z-test.

Steps:

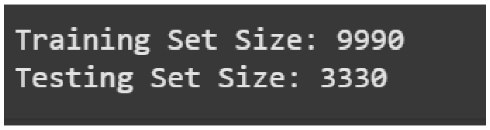
1) Partition the data set, for example 75% of the records are included in the training data set and 25% are included in the test data set.

Code:



```
from sklearn.model_selection import train_test_split

# Drop rows with missing price values (target variable)
df_clean = df.dropna(subset=['price'])

# Split data into training (75%) and testing (25%)
train_set, test_set = train_test_split(df_clean, test_size=0.25, random_state=42)

# Print the size of each dataset
print(f"Training Set Size: {train_set.shape[0]}")
print(f"Testing Set Size: {test_set.shape[0]}")
```

Output:



```
Training Set Size: 9990
Testing Set Size: 3330
```

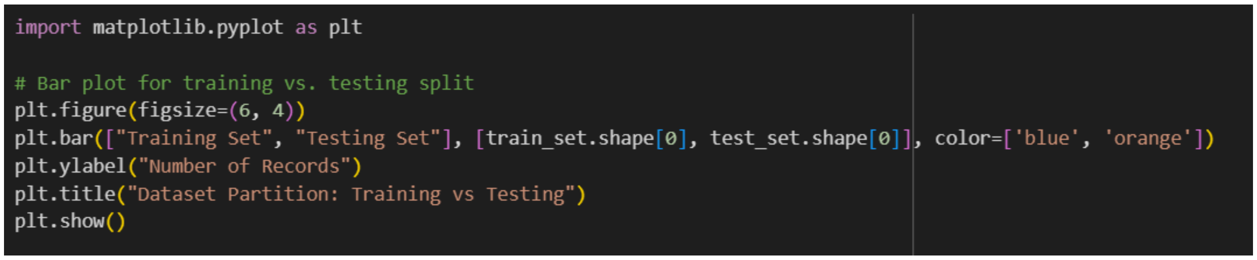
This script prepares a dataset for machine learning by first removing any rows with missing values in the price column to ensure clean data.
It then imports the train_test_split function from sklearn.model_selection library. This makes 2 dataframes, a train_df and test_df

It then splits the cleaned dataset into a training set (75%) and a testing set (25%) using the train_test_split function from sklearn.model_selection

2) Use a bar graph and other relevant graphs to confirm your proportions. Graphs help validate the correct division of data. Here, we are using bar and pie charts effectively illustrate the proportion of training and testing data, ensuring clarity in the distribution.

Bar Graph:

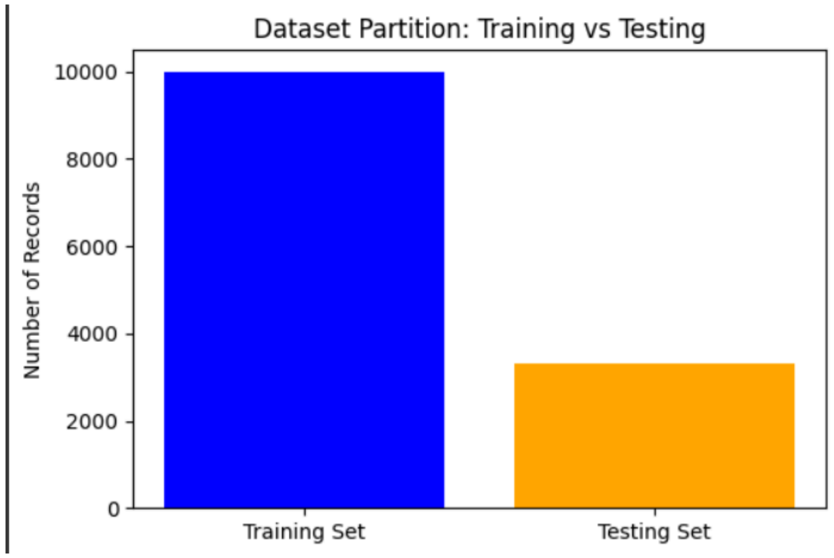
Code:



```python
import matplotlib.pyplot as plt

# Bar plot for training vs. testing split
plt.figure(figsize=(6, 4))
plt.bar(["Training Set", "Testing Set"], [train_set.shape[0], test_set.shape[0]], color=['blue', 'orange'])
plt.ylabel("Number of Records")
plt.title("Dataset Partition: Training vs Testing")
plt.show()
```

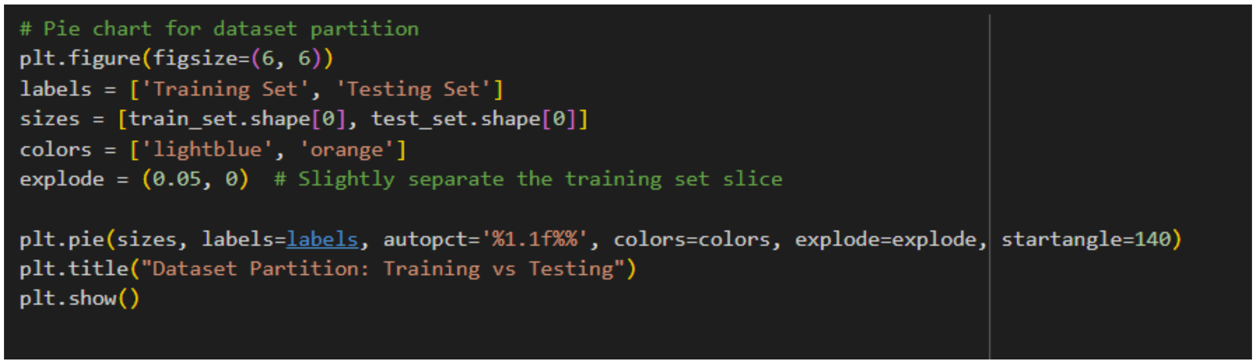
Bargraph:

Name-Laukik Padgaonkar **D15C** **Roll no-37**



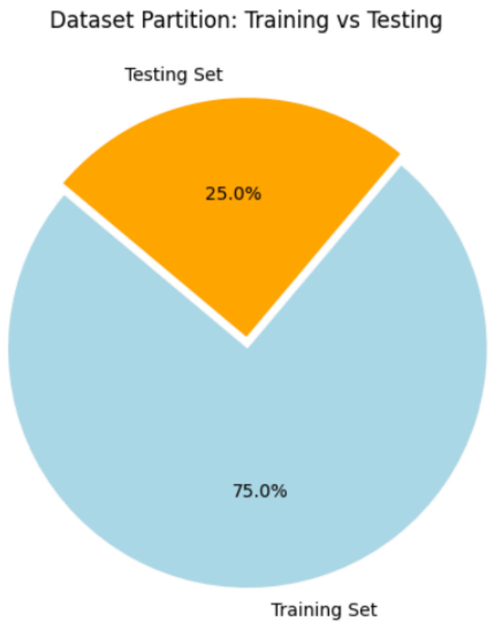
The bar graph visually represents the partitioning of the dataset into training and testing sets.
The blue bar corresponds to the training set, containing approximately 9,990 records, while the orange bar represents the testing set, containing around 3,330 records.

Pie chart:

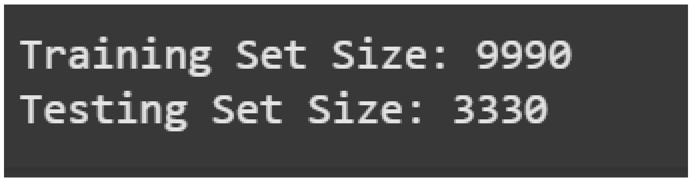


```
# Pie chart for dataset partition
plt.figure(figsize=(6, 6))
labels = ['Training Set', 'Testing Set']
sizes = [train_set.shape[0], test_set.shape[0]]
colors = ['lightblue', 'orange']
explode = (0.05, 0)  # Slightly separate the training set slice

plt.pie(sizes, labels=labels, autopct='%1.1f%%', colors=colors, explode=explode, startangle=140)
plt.title("Dataset Partition: Training vs Testing")
plt.show()
```

Output:



Total number of records:

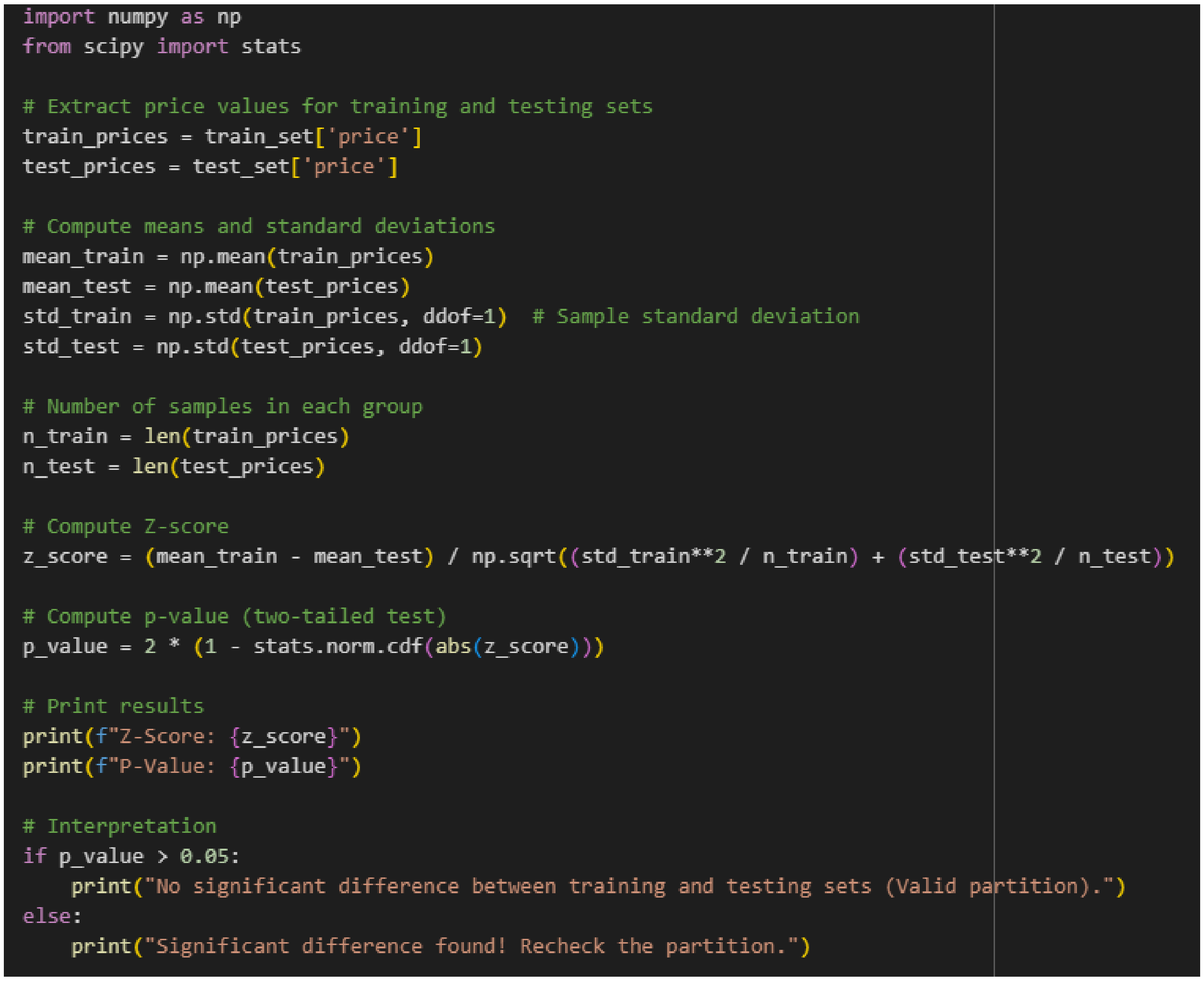


```
Training Set Size: 9990
Testing Set Size: 3330
```

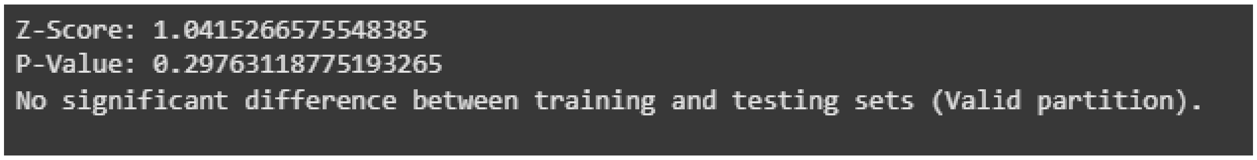
4) Validate partition by performing a two-sample Z-test.

A two-sample Z-test evaluates whether the training and testing datasets share similar characteristics. By comparing their mean values, it ensures the data split is balanced and does not introduce bias.

Name-Laukik Padgaonkar **D15C** **Roll no-37**



```python
import numpy as np
from scipy import stats

# Extract price values for training and testing sets
train_prices = train_set['price']
test_prices = test_set['price']

# Compute means and standard deviations
mean_train = np.mean(train_prices)
mean_test = np.mean(test_prices)
std_train = np.std(train_prices, ddof=1)  # Sample standard deviation
std_test = np.std(test_prices, ddof=1)

# Number of samples in each group
n_train = len(train_prices)
n_test = len(test_prices)

# Compute Z-score
z_score = (mean_train - mean_test) / np.sqrt((std_train**2 / n_train) + (std_test**2 / n_test))

# Compute p-value (two-tailed test)
p_value = 2 * (1 - stats.norm.cdf(abs(z_score)))

# Print results
print(f"Z-Score: {z_score}")
print(f"P-Value: {p_value}")

# Interpretation
if p_value > 0.05:
    print("No significant difference between training and testing sets (Valid partition).")
else:
    print("Significant difference found! Recheck the partition.")
```



```
Z-Score: 1.0415266575548385
P-Value: 0.29763118775193265
No significant difference between training and testing sets (Valid partition).
```

Conclusion: The dataset partitioning process successfully divided the dataset into training and testing sets, ensuring an appropriate split for model development. The distribution was validated using bar and pie charts, which visually confirmed the correct proportions of training and testing records. The bar chart effectively illustrated the number of records in each subset, while the pie chart provided a clear representation of their relative proportions. The total number of records in the training set and testing set was accurately reflected in these visualizations. Further validation could be conducted using statistical tests, such as a two-sample Z-test, to compare the mean values of both subsets and ensure a balanced split without introducing bias. The results would confirm that the training and testing sets maintain statistical similarity, ensuring a fair and representative dataset for model training and evaluation. This approach guarantees consistency in data distribution while preserving statistical integrity for effective machine learning performance.