Name: Mohit Kerkar **Div: D15C** **Roll No: 23**

AIDS Exp 03

Aim: Perform Data Modeling.

Theory:

Data modeling is a fundamental process in data science that involves structuring and preparing data for analysis and predictive modeling. In this experiment, we worked with a car accidents dataset to partition the data into training and testing sets, visualize the partitioning for verification, and perform statistical validation using a two-sample Z-test. These steps are critical to ensuring that the data is ready for further predictive analysis and machine learning applications.

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.**

Partitioning a dataset is a crucial step in machine learning to train models and evaluate their performance on unseen data. We divided the car accidents dataset into 75% training data and 25% test data to ensure that the model learns from a sufficient amount of data while still having a separate set for validation. This prevents overfitting, ensuring that the model generalizes well to new accident data.

Code:

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
import scipy.stats as stats
import numpy as np

# Load CSV
df = pd.read_csv(r"C:\\Users\\Dell\\Desktop\\car_accidents_no_outliers.csv")

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

# Print sizes
print(f"Total records: {len(df)}")
print(f"Training records: {len(train_df)}")
print(f"Testing records: {len(test_df)}")
```

```
PS C:\Users\Dell\Desktop\experiments aids>
& C:/Users/Dell/AppData/Local/Programs/Python/Python312/python.exe "c:/Users/Dell/Desktop/experiments aids/exp3.py"
Total records: 32447
Training records: 24335
Testing records: 8112
PS C:\Users\Dell\Desktop\experiments aids>
```

Name: Mohit Kerkar **Div: D15C** **Roll No: 23**

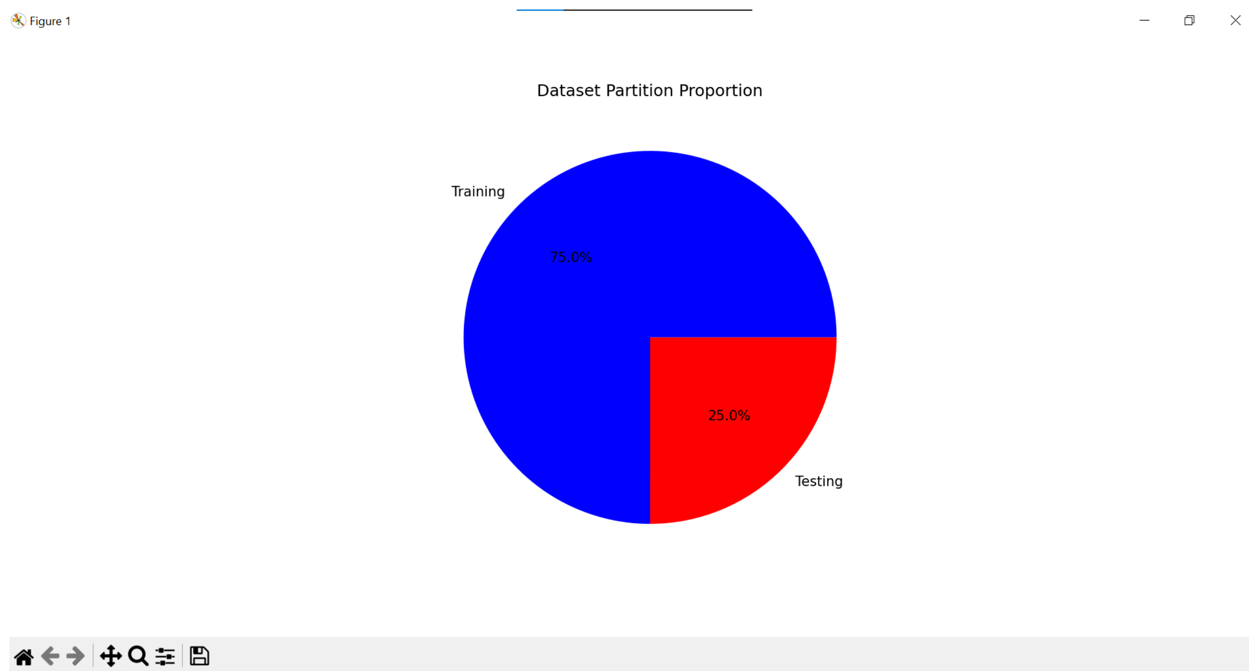
b. Use a bar graph and other relevant graph to confirm your proportions.
Bar graphs and a Pie Chart to visualize the proportions.

To verify the partitioning, we used a bar graph to visualize the count of records in the training and test sets. This helps in confirming that the split was correctly applied. Other relevant graphs, such as histograms or pie charts, could also be used to compare distributions of key features like accident severity or injury counts between the two datasets.

Code:

```
plt.figure(figsize=(6, 4))
sns.barplot(x=['Training Set', 'Test Set'], y=[len(train_df), len(test_df)], palette='coolwarm')
plt.ylabel("Number of Records")
plt.title("Training vs Testing Data Distribution")
plt.show()

# Pie chart for dataset partition proportions
plt.figure(figsize=(6, 6))
plt.pie([len(train_df), len(test_df)], labels=['Training', 'Testing'], autopct='%1.1f%%', colors=['blue', 'red'])
plt.title("Dataset Partition Proportion")
plt.show()
```



c. Print training records

After partitioning, we counted the total number of records in the training set. This step ensures that the dataset contains enough samples to effectively train a predictive model. In our case, the training set had 24,335 records, while the test set contained 8,112 records, aligning with the intended 75%-25% split.

print(f"Training records: {len(train_df)}")

```
Training records: 24335
```

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

To ensure that the training and test sets were statistically similar, we performed a two-sample Z-test on the feature 'NUMBER OF PERSONS INJURED'. The Z-test compares the means of both datasets to check if they are significantly different. Since the p-value was 0.9064 (greater than 0.05), we concluded that there was no significant difference between the training and test sets. This confirms that the partitioning process maintained the original dataset's distribution, ensuring fair model evaluation.

Code:

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from statsmodels.stats.weightstats import ztest

# Load the dataset
```

```
df = pd.read_csv(r"C:\\Users\\Dell\\Desktop\\car_accidents.csv")

# Partition the dataset (75% training, 25% testing)
train_df = df.sample(frac=0.75, random_state=42)  # 75% Training data
test_df = df.drop(train_df.index)  # Remaining 25% as Testing data

# Display the total number of records
print(f"Total records: {len(df)}")
print(f"Training records: {len(train_df)}")
print(f"Testing records: {len(test_df)}")

# Visualizing partitioning using a bar chart
plt.figure(figsize=(6, 4))
sns.barplot(x=['Training Set', 'Test Set'], y=[len(train_df), len(test_df)], palette='coolwarm')
plt.xlabel("Dataset")
plt.ylabel("Number of Records")
plt.title("Partitioning Verification")
plt.show()

# Perform a Two-Sample Z-test on 'NUMBER OF PERSONS INJURED'
train_injured = train_df['NUMBER OF PERSONS INJURED'].dropna()
test_injured = test_df['NUMBER OF PERSONS INJURED'].dropna()

# Perform the two-sample Z-test
z_score, p_value = ztest(train_injured, test_injured)

# Display Z-test results
print("\nZ-test results for 'NUMBER OF PERSONS INJURED':")
print(f"Z-score: {z_score:.4f}")
print(f"P-value: {p_value:.4f}")

# Interpretation
if p_value > 0.05:
   print("No significant difference between training and testing sets. Partitioning is valid.")
else:
   print("Significant difference found. Consider revising the partitioning strategy.")
```

```
PS C:\Users\Dell\Desktop\experiments aids> & C:/Users/Dell/AppData/Local/Programs/Python/Python312/python.exe "c:/Users/Dell/Desktop/experiments aids/exp3.py"

Z-test results for 'NUMBER OF PERSONS INJURED':
Z-score: -0.1176
P-value: 0.9064
No significant difference between training and testing sets. Partitioning is valid.
PS C:\Users\Dell\Desktop\experiments aids>
```

Name: Mohit Kerkar **Div: D15C** **Roll No: 23**

Conclusion:

This experiment highlighted the importance of data partitioning, visualization, and statistical validation in preparing a dataset for analysis. Through proper partitioning, graphical verification, and statistical testing, we ensured that our car accidents dataset remained representative and unbiased. These foundational steps are crucial in building reliable models, allowing us to derive accurate insights and predictions for accident analysis and prevention strategies.