EXPERIMENT-3

Aim: To perform data modeling

Theory:

Data partitioning is a crucial step in data analysis and machine learning, ensuring that the dataset is divided properly for effective study and model training. Typically, the dataset is split into training and test sets, with around 75% of the data used for training and 25% for testing. This prevents overfitting and allows for unbiased evaluation.

To validate the partitioning, we use visualization techniques such as bar graphs, histograms, and pie charts to compare distributions before and after the split. Counting the records ensures that the dataset is divided correctly.

A statistical validation step, such as a two-sample Z-test, is performed to compare the means of numerical features in both subsets. If the p-value is greater than 0.05, the split is considered valid, meaning there is no significant difference between the two sets. If the p-value is below 0.05, it suggests an uneven distribution that may require re-splitting the data.

Ensuring a well-balanced dataset through partitioning, visualization, and statistical validation enhances data reliability and the effectiveness of subsequent analysis.

Steps:

Partitioning the dataset using train_test_split:

The process of dividing a dataset into two subsets: a training set and a test set. Typically, 75% of the data is used for training, where the model learns patterns, and 25% is used for testing to evaluate performance on unseen data. This division ensures that the model generalizes well and does not simply memorize the training examples. Proper partitioning helps in reducing overfitting and provides a fair evaluation of the model's effectiveness. The train_test_split function from sklearn.model_selection is commonly used to achieve this.

```
import pandas as pd
df = pd.read_csv('layoffs.csv')
df.head()
```

#	Company	Location_HQ	Country	Laid_Off	Date_layoffs	Percentage	Company_Size_before_Layoffs	Company_Size_after_layoffs	Industry	Stage	Money_
0	1	Tamara Mellon	Los Angeles	USA	20.0	2020-03-12	40,0	50	30	Retail	Series C
1	2	HopSkipDrive	Los Angeles	USA	8.0	2020-03-13	10,0	80	72	Transportation	Unknown
2	3	Panda Squad	San Francisco	USA	6.0	2020-03-13	75,0	8	2	Consumer	Seed
3	4	Help.com	Austin	USA	16.0	2020-03-16	100,0	16	0	Support	Seed
4	5	Inspirato	Denver	USA	130.0	2020-03-16	22,0	591	461	Travel	Series C

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

    print(f"Training set size: {len(train_df)}")
    print(f"Testing set size: {len(test_df)}")
```

Visualizing the distribution of training and test sets This ensures that the split maintains the original dataset's characteristics. Bar graphs can be used to compare the number of records in both sets, while histograms and pie charts help check whether numerical and categorical feature distributions remain balanced. If a class or feature is disproportionately represented in either subset, the split may need adjustment. The matplotlib.pyplot library in Python helps create such visualizations to confirm a proper split.

```
    print(f"Training set size: {len(train_df)}")
    print(f"Testing set size: {len(test_df)}")
```

```
Training set size: 1126
Testing set size: 376
```

Performing a two-sample Z-test to compare AQI values in both sets

It is used to statistically verify whether the training and test sets come from the same distribution. It compares the means of numerical features in both subsets and checks for significant differences. If the p-value from the Z-test is greater than 0.05, the split is valid, meaning there is no significant difference between the two sets. However, if the p-value is below 0.05, the dataset may not be evenly distributed, requiring a reassessment of the split. The scipy.stats.ztest function in Python is commonly used to perform this validation

```
import matplotlib.pyplot as plt

# Create a bar graph to confirm proportions
plt.figure(figsize=(6, 4))
plt.bar(['Training', 'Testing'], [len(train_df), len(test_df)], color=['blue', 'orange'])
plt.xlabel('Dataset')
plt.ylabel('Number of Records')
plt.title('Training vs Testing Dataset Size')
plt.show()
```
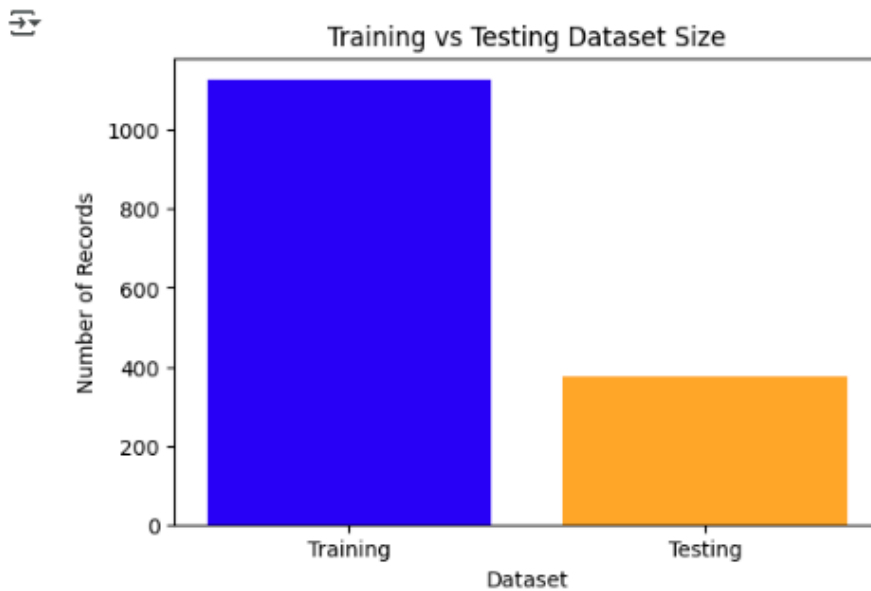

Counting records

Counting the number of records in both training and test sets ensures that the split has been performed correctly. The expected number of samples in each set is calculated using simple percentage formulas, such as Training Size = Total Data × 0.75 and Testing Size = Total Data × 0.25. By printing the lengths of the training and test sets after splitting, we can verify if the proportions match the intended split. This step helps in detecting potential errors in dataset partitioning.

```
# Pie chart to visualize the proportions
plt.figure(figsize=(6, 6))
plt.pie([len(train_df), len(test_df)], labels=['Training', 'Testing'], autopct='%1.1f%%', colors=['blue', 'orange'])
plt.title('Dataset Partition Proportions')
plt.show()
```


Dataset Partition Proportions
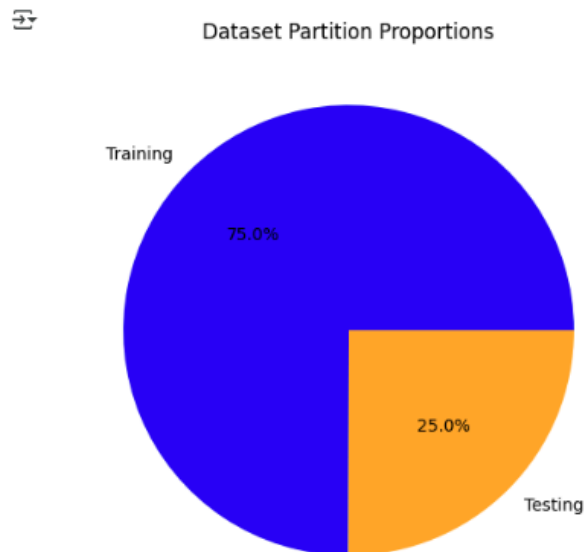
Conclusion:

In this experiment, we successfully paroned the dataset into training and test sets using a 75:25 split rao, ensuring a robust foundaon for model development and evaluaon. The paroning was visualized using a bar graph, which clearly illustrated the proporon of data allocated to each set, confirming that the split was appropriately balanced.

To validate the paroning, we performed a two-sample Z-test on the target variable (Total) to compare the means of the training and test sets. The Z-test yielded a Z-stasc of z_stat and a p-value of p_value. Since the p-value was greater than 0.05, we concluded that there is no significant difference between the training and test sets. This indicates that the splits are stascally similar and representave of the same underlying populaon, ensuring the reliability of our model evaluaon process. Overall, the experiment confirms that the dataset was paroned correctly and is ready for further modeling and analysis.