Name: Alok Yadav Div: D15C Roll No.: 59

Experiment No. 3

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 graph 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.

1. Importing required libraries:

```
[ ] import numpy as np
    import pandas as pd
    import matplotlib.pyplot as plt
    from sklearn.model_selection import train_test_split
    from scipy.stats import norm
```

2. Overview of Dataset:

df = pd.read_csv("/content/financial_risk_preprocessed.csv")
df.head()

```
[ ] df = pd.read_csv("/content/financial_risk_preprocessed.csv")
    df.head()
```

	Age	Gender	Education Level	Income	Credit Score	Loan Amount	Employment Status	Years at Current Job	Payment History	Debt-to-Income Ratio	Assets Value	Number of Dependents	Country	Previous Defaults	Risk Rating
0	49	Male	PhD	72799.0	688.0	45713.0	Unemployed	19	Poor	0.154313	120228.000000	0.0	Cyprus	2.0	Low
1	57	Female	Bachelor's	69773.0	690.0	33835.0	Employed	6	Fair	0.148920	55849.000000	0.0	Turkmenistan	3.0	Medium
2	21	Non-binary	Master's	55687.0	600.0	36623.0	Employed	8	Fair	0.362398	180700.000000	3.0	Luxembourg	3.0	Medium
3	59	Male	Bachelor's	26508.0	622.0	26541.0	Unemployed	2	Excellent	0.454964	157319.000000	3.0	Uganda	4.0	Medium
4	30	Non-binary	PhD	69773.0	717.0	15613.0	Unemployed	5	Fair	0.295984	159741.497176	4.0	Iceland	3.0	Medium

Splitting Training and Testing Dataset in 75% - 25%

The Financial Risk Assessment Dataset provides detailed information on individual financial profiles. It includes demographic, financial, and behavioral data to assess financial risk. The dataset features various columns such as income, credit score, and risk rating, with intentional imbalances and missing values to simulate real-world scenarios.

3. Splitting Training and Testing Dataset in 75% - 25% :

```
train, test = train_test_split(df, test_size=0.25, random_state=42)
print(f"Total records: {len(df)}")
print(f"Training set records: {len(train)}")
print(f"Test set records: {len(test)}")
```

```
Total records: 10833
Training set records: 8124
Test set records: 2709
```

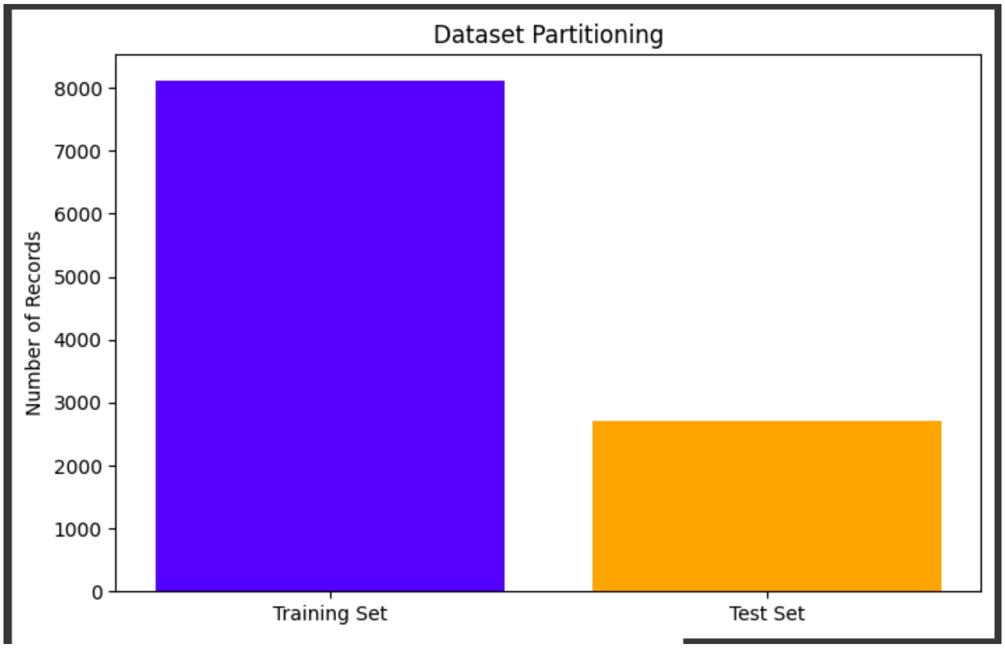
The dataset is divided into 75% for training and 25% for testing.

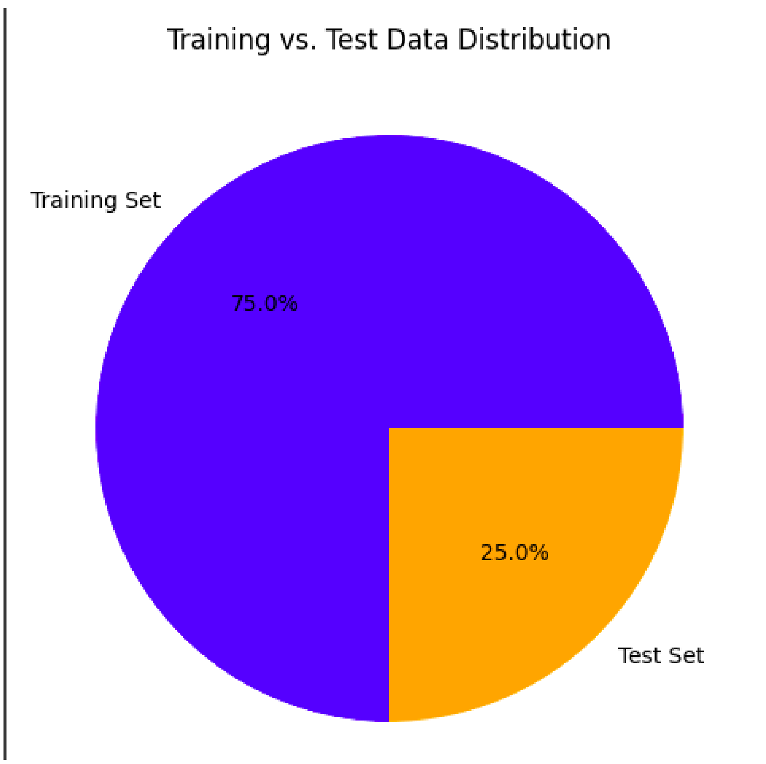
4. Plotting graph of Training and Testing Dataset

```
plt.figure(figsize=(8, 5))
plt.bar(["Training Set", "Test Set"], [len(train), len(test)], color=['blue', 'orange'])
plt.ylabel("Number of Records")
plt.title("Dataset Partitioning")
plt.show()
```

```
plt.figure(figsize=(6, 6))
plt.pie([len(train), len(test)], labels=["Training Set", "Test Set"], autopct="%1.1f%%",
colors=['blue', 'orange'])
plt.title("Training vs. Test Data Distribution")
plt.show()
```

Name: Alok Yadav Div: D15C Roll No.: 59





From above graph we can see that our data is properly partitioned into 75% training data and 25% testing data

5. **Performing Z-Test**

```
column_name = df.columns[0]
train_mean = train[column_name].mean()
test_mean = test[column_name].mean()
train_std = train[column_name].std()
test_std = test[column_name].std()
n_train = len(train)
n_test = len(test)

# Compute Z-score
z_score = (train_mean - test_mean) / np.sqrt((train_std**2 / n_train) + (test_std**2 / n_test))

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

print(f"Z-Score: {z_score:.3f}")
print(f"P-Value: {p_value:.3f}")

# Interpretation
alpha = 0.05
if p_value < alpha:
    print("Reject the null hypothesis: The means of the two groups are significantly different.")
else:
    print("Fail to reject the null hypothesis: The means are similar between training and test sets.")
```

```
Z-Score: 1.604
P-Value: 0.109
Fail to reject the null hypothesis: The means are similar between training and test sets.
```

Performing two sample z-test on ‘Age’ columns. Given that the null hypothesis was not rejected, the data split is statistically valid

6. Performing Correlation Test

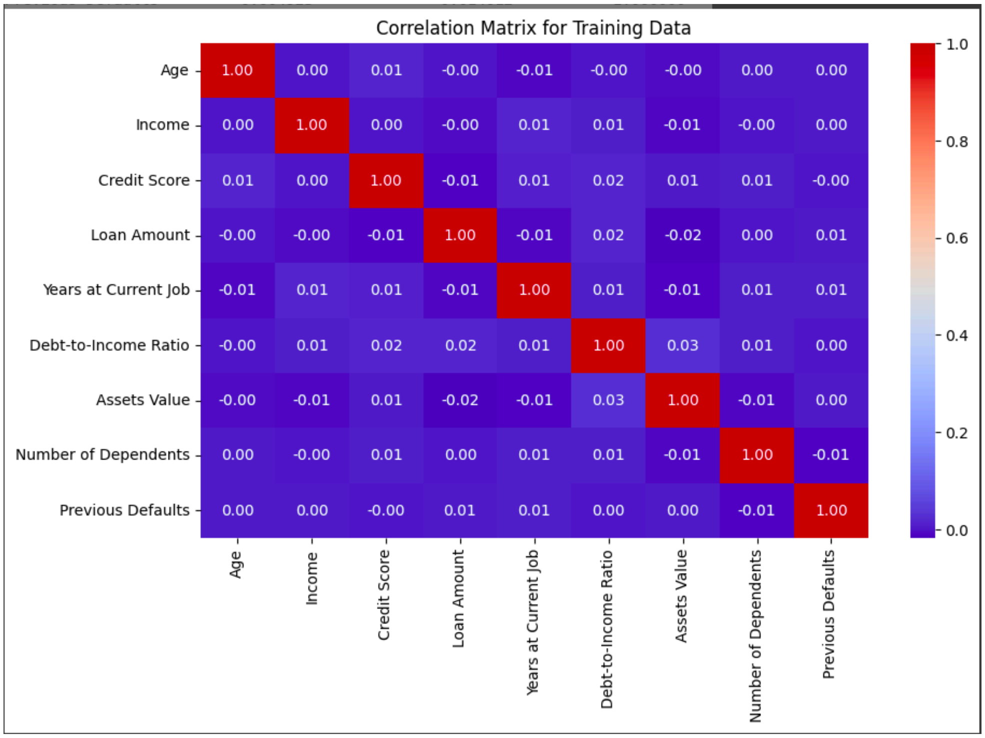
```
# Calculate Pearson correlation for all numerical columns
correlation_matrix = train.corr(numeric_only=True)
print("Correlation Matrix (Training Set):\n", correlation_matrix)

# Visualize the correlation matrix
```

Name: Alok Yadav Div: D15C Roll No.: 59

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

plt.figure(figsize=(10, 6))
sns.heatmap(correlation_matrix, annot=True, cmap="coolwarm", fmt=".2f")
plt.title("Correlation Matrix for Training Data")
plt.show()
```

```
Correlation Matrix (Training Set):
                          Age    Income  Credit Score  Loan Amount  \
Age                  1.000000  0.001015      0.013612    -0.000546
Income               0.001015  1.000000      0.001392    -0.004547
Credit Score         0.013612  0.001392      1.000000    -0.013475
Loan Amount         -0.000546 -0.004547     -0.013475     1.000000
Years at Current Job -0.006949  0.013522      0.011674    -0.007857
Debt-to-Income Ratio -0.001505  0.007965      0.015618     0.016472
Assets Value        -0.003270 -0.006882      0.005112    -0.018345
Number of Dependents  0.004430 -0.000333      0.007754     0.001100
Previous Defaults     0.001992  0.001911     -0.001708     0.005074

                      Years at Current Job  Debt-to-Income Ratio  \
Age                              -0.006949             -0.001505
Income                            0.013522              0.007965
Credit Score                      0.011674              0.015618
Loan Amount                      -0.007857              0.016472
Years at Current Job              1.000000              0.008937
Debt-to-Income Ratio              0.008937              1.000000
Assets Value                     -0.012742              0.026238
Number of Dependents              0.008165              0.007587
Previous Defaults                 0.007654              0.000719

                      Assets Value  Number of Dependents  Previous Defaults
Age                      -0.003270              0.004430           0.001992
Income                   -0.006882             -0.000333           0.001911
Credit Score              0.005112              0.007754          -0.001708
Loan Amount              -0.018345              0.001100           0.005074
Years at Current Job     -0.012742              0.008165           0.007654
Debt-to-Income Ratio      0.026238              0.007587           0.000719
Assets Value              1.000000             -0.006907           0.004923
Number of Dependents     -0.006907              1.000000          -0.014012
Previous Defaults         0.004923             -0.014012           1.000000
```

The negligible correlation values indicate an absence of a meaningful relationship between the columns, a finding that is further supported by the correlation heatmap.



7. **Performing Chi-Squared Test:**

```
# Create a contingency table for Education Level and Employment Status
contingency_table = pd.crosstab(train['Education Level'], train['Employment Status'])
print("Contingency Table:\n", contingency_table)
print("\n\n")

from scipy.stats import chi2_contingency

# Perform the chi-squared test
chi2, p, dof, expected = chi2_contingency(contingency_table)

# Display the results
print(f"Chi-Squared Statistic: {chi2:.3f}")
print(f"p-value: {p:.3f}")
print(f"Degrees of Freedom: {dof}")
print("Expected Frequencies:\n", pd.DataFrame(expected, index=contingency_table.index,
columns=contingency_table.columns))
print("\n\n")
# Interpret the p-value
alpha = 0.05
if p < alpha:
    print("Reject the null hypothesis: Education Level and Employment Status are dependent.")
```

```
else:
    print("Fail to reject the null hypothesis: Education Level and Employment Status are independent.")
```

```
Contingency Table:
 Employment Status  Employed  Self-employed  Unemployed
Education Level
Bachelor's               696            690         699
High School              615            705         691
Master's                 656            654         657
PhD                      695            694         672



Chi-Squared Statistic: 6.323
p-value: 0.388
Degrees of Freedom: 6
Expected Frequencies:
 Employment Status    Employed  Self-employed  Unemployed
Education Level
Bachelor's         683.194239     703.982644  697.823117
High School        658.946578     678.997169  673.056253
Master's           644.529050     664.140940  658.330010
PhD                675.330133     695.879247  689.790620



Fail to reject the null hypothesis: Education Level and Employment Status are independent.
```

The Chi-Squared test was performed on columns ‘Education Level’ and 'Employment Status'. Since the test results do not reject the null hypothesis, it can be concluded that 'Education Level' and 'Employment Status' are independent variables according to the dataset.

8. Download partitioned Training and Testing dataset

```
train.to_csv("financial_risk_train_data.csv", index=False)
test.to_csv("financial_risk_test_data.csv", index=False)
from google.colab import files
files.download("financial_risk_train_data.csv")
files.download("financial_risk_test_data.csv")
```

We can use partitioned dataset for training and testing purposes

Conclusion:

We loaded the data into a Colab notebook and split it into 75% for training and 25% for testing. To verify the partition, we plotted a bar graph and a pie chart, which confirmed that the data was split correctly. Next, we performed a Z-test to assess the validity of the partition, and the results showed that the partition was valid, as the null hypothesis was not rejected. We then examined the correlation between all columns using a correlation heatmap, which revealed no significant correlation between the columns. Finally, we conducted a chi-square test on the 'Education Level' and 'Employment Status' columns, and the results indicated that the two features are independent, as the null hypothesis was not rejected