Data Visualization and Pre-processingPerform Below Tasks to complete the assignment:-Tasks:-

1. Download the dataset: Dataset

2. Load the dataset.

3. Perform Below Visualizations.
- Univariate Analysis
- Bi - Variate Analysis
- Multi - Variate Analysis

4. Perform descriptive statistics on the dataset.

5. Handle the Missing values.

6. Find the outliers and replace the outliers

7. Check for Categorical columns and perform encoding.

8. Split the data into dependent and independent variables.

9. Scale the independent variables

10. Split the data into training and testing

1. Univariate Analysis:

Data consists of only one variable (only x value).

a. Line Plots / Bar Charts
b. Histograms
c. Box Plots
d. Count Plots
e. Descriptive Statistics techniques
f. Violin Plot

2. Bivariate Analysis:

When we talk about bivariate analysis, it means analyzing 2 variables. Since we know there are numerical and categorical variables, there is a way of analyzing these variables as shown below:

i) Numerical & Numerical

a. Scatterplot
b. Line plot
c. Heatmap for correlation
d. Joint plot

ii) Numerical & Categorical

a. Bar chart
b. Violin plot
c. Categorical box plot
d. Swarm plot

iii) Categorical & Categorical

a. Bar chart
b. Grouped bar chart
c. Point plot

3. Multivariate Analysis:
In the case of 3 or more variables

a. Pair Plot

The simplest case we will consider is to find descriptive statistics for the entire dataset. In this case, no variable specification is required and we must simply specify the dataset name. For example, consider finding descriptive statistics using the demographics and crime statistics in Detroit stored in the SAS dataset detriot.sas7bdat

```
//Compute statistics for all variables in the dataset
fname = getGAUSSHome() $+ "examples/detroit.sas7bdat";

//The 'call' keyword disregards return values from the function
call dstatmt(fname);
```

The output from the above code is:

```
---------------------------------------------------------------------------------------------------
Variable                    Mean     Std Dev      Variance     Minimum      Maximum      Valid Missing
---------------------------------------------------------------------------------------------------


year                  1967.0000      3.8944       15.1667   1961.0000   1973.0000        13    0
ft_police               304.5115     46.8117     2191.3312    260.3500    390.1900        13    0
unemployment               5.7923      2.3592        5.5658      3.2000     11.0000        13    0
manufacture_employ        556.4462     49.8222     2482.2477    455.5000    613.5000        13
0
gun_license             537.5069    316.4151   100118.5406    156.4100   1131.2100        13    0
gun_registration         545.6592    311.0316    96740.6634    180.4800   1029.7500        13    0
```

```
homicide_clearance        81.4462     12.6592     160.2560     58.9000     94.4000       13    0
num_white_males       452507.5385  64568.1239 4169042623.43 359647.0000 558724.0000
13    0
non_manufacture_employ     673.9231     94.7734     8981.9969    538.1000    819.8000       13
0
govt_employ             185.7692     37.0362     1371.6790    133.9000    230.9000       13    0
hourly_earn               3.9477      0.9666       0.9342      2.9100      5.7600       13    0
weekly_earn             169.9708     42.5112     1807.2053    117.1800    258.0500       13    0
homicide                 25.1269     16.3854      268.4825      8.5200     52.3300       13    0
accident_death             46.9231      5.1396      26.4155     39.1700     55.0500       13    0
assault                 311.9500     73.0912     5342.3166    217.9900    473.0100       13    0
```

```
    missing_val_count_by_column = (data.isnull().sum())
print(missing_val_count_by_column[missing_val_count_by_column > 0
```

Most libraries (including scikit-learn) will give you an error if you try to build a model using data with missing values. So you'll need to choose one of the strategies below.

```
        # Import required libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

# Reading the data
df = pd.read_csv("data_out.csv")
print(df.shape)
print(df.info())
python
Output:

   (600, 6)
   <class 'pandas.core.frame.DataFrame'>
   RangeIndex: 600 entries, 0 to 599
   Data columns (total 6 columns):
   Income            600 non-null int64
   Loan_amount        600 non-null int64
   Term_months        600 non-null int64
   Credit_score       600 non-null int64
   approval_status    600 non-null int64
   Age               600 non-null int64
   dtypes: int64(6)
   memory usage: 28.2 KB
```

```
    None
```

```
# import required libraries
import pandas as pd
import numpy as np
# creating initial dataframe
bridge_types = ('Arch','Beam','Truss','Cantilever','Tied Arch','Suspension','Cable')
bridge_df = pd.DataFrame(bridge_types, columns=['Bridge_Types'])
# converting type of columns to 'category'
bridge_df['Bridge_Types'] = bridge_df['Bridge_Types'].astype('category')
# Assigning numerical values and storing in another column
bridge_df['Bridge_Types_Cat'] = bridge_df['Bridge_Types'].cat.codes
bridge_df
```

Using sci-kit learn library approach:

```
    import pandas as pd
df = pd.read_csv('household_data.csv')
print(df)
pd
```

Output:

```
  Item_Category         Gender          Age     Salary  Purchased
0  Fitness       Male    20       30000  Yes
1  Fitness       Female           50       70000  No
2  Food                   Male    35       50000  Yes
3  Kitchen       Male    22       40000  No
4  Kitchen       Female           30       35000
```

```
        import pandas as pd

from sklearn.preprocessing import StandardScaler

# Read Data from CSV

data = read_csv('Geeksforgeeks.csv')
data.head()
```

```
# Initialise the Scaler

scaler = StandardScaler()

# To scale data
scaler.fit(data)
```

Code Intelligence

Type search here. Hit enter to submit or escape to close.

Fuzzing As Easy As Unit Testing

CI Fuzz CLI is an open-source solution that lets you run feedback-based fuzz tests from your command line. Every developer can use it to find bugs and vulnerabilities with three simple commands.

```
# Initialize fuzzing

$ cifuzz init

# Create your first fuzz test

$ cifuzz create my_fuzz_test

# Run fuzz test and find bugs

$ cifuzz run my_fuzz_test
```