stats exp 4

Okay, I can help you write a full experiment report about data preprocessing, focusing on handling missing values and understanding data characteristics.

Experiment Number: 4

AIM: To perform data preprocessing techniques on a real-world dataset, specifically focusing on identifying, analyzing, and handling missing values, as well as understanding the different types of variables and the dimensions of the dataset.

Objectives:

- To learn how to identify and analyze missing values in a dataset.
- To understand and apply various techniques for handling missing data, including deletion and imputation methods.
- To differentiate between numerical and categorical variables and their subtypes.
- To determine the dimensions of a dataset and understand its structure.

Course Outcomes: CO3 (This will likely relate to data preprocessing and exploration within your course.)

Resource/Tools: Jupyter Notebook/Python with Pandas and Scikit-learn libraries.

Procedure/Algorithm/Code:

1. **Dataset Selection and Loading:**
    - Choose an open-source dataset with a significant number of missing values. (Examples: Titanic dataset, House Prices dataset from Kaggle)
    - Load the dataset into a Pandas DataFrame:

```
import pandas as pd
data = pd.read_csv('your_dataset.csv')  # Replace
'your_dataset.csv'
```

2. **Initial Data Exploration:**
    - Display the first few rows of the data using `data.head()`.
    - Get information about the columns, their data types, and the number of non-null values using `data.info()`.
    - Generate descriptive statistics (count, mean, std, etc.) using `data.describe()`.
3. **Missing Value Analysis:**
    - Calculate the number of missing values for each column:

```
print(data.isnull().sum())
```

- (Optional) Visualize the pattern of missing data using the `missingno` library:

```
import missingno as msno
msno.matrix(data)
msno.heatmap(data)  # To visualize correlations between missingness
plt.show()
```

4. **Handling Missing Values:**
   - **Deletion:**
     - If a small percentage of rows have missing values, consider removing them:

```
data.dropna(inplace=True)  # Removes rows with any missing values
```

     - If a particular column has a high percentage of missing values and is not crucial, consider dropping it:

```
data.drop('column_name', axis=1, inplace=True)
```

   - **Imputation:**
     - **Simple Imputation:** Replace missing values with a constant, mean, median, or mode:

```
from sklearn.impute import SimpleImputer
imputer = SimpleImputer(strategy='mean')  # Or 'median', 'most_frequent'
data['numerical_column'] = imputer.fit_transform(data[['numerical_column']])
```

     - **KNN Imputation:** Use the values of the 'k' nearest neighbors to fill in missing values:

```
from sklearn.impute import KNNImputer
imputer = KNNImputer(n_neighbors=5)
data['numerical_column'] = imputer.fit_transform(data[['numerical_column']])
```

     - **Regression Imputation (Example):**

```
from sklearn.linear_model import LinearRegression
lr = LinearRegression()
testdf = data[data['Age'].isnull()]  # Assuming 'Age' has
missing values
traindf = data[data['Age'].notnull()]
y = traindf['Age']
traindf.drop("Age", axis=1, inplace=True)
lr.fit(traindf, y)
testdf.drop("Age", axis=1, inplace=True)
pred = lr.predict(testdf)
testdf['Age'] = pred
data = pd.concat([traindf, testdf], axis=0)  # Combine the
data back
```

5. **Variable Type Identification:**
   - Identify numerical and categorical variables:

```
numerical_cols =
data.select_dtypes(include=np.number).columns.tolist()
categorical_cols = data.select_dtypes(include=
['object']).columns.tolist()
print("Numerical Columns:", numerical_cols)
print("Categorical Columns:", categorical_cols)
```

   - Further classify categorical variables as nominal or ordinal based on their nature.
6. **Check Data Dimensions:**
   - Determine the number of rows and columns:

```
print(data.shape)  # Output: (number of rows, number of columns)
```

Output:

- The output will include:
  - First few rows of the dataset.
  - Column information, data types, and missing value counts.
  - Descriptive statistics of the data.
  - (Optional) Visualizations of missing data patterns.
  - Output from imputation methods.
  - Lists of numerical and categorical columns.
  - Dimensions of the DataFrame.

Result:

- The experiment successfully identified and handled missing values in the chosen dataset using appropriate techniques.
- The analysis provided insights into the distribution of missing data and the effectiveness of different imputation strategies.
- The types of variables and the dimensions of the dataset were clearly identified, contributing to a better understanding of the data structure.

Learning Outcomes:

- Gained practical experience in identifying, analyzing, and addressing missing data in a real-world dataset.
- Developed a deeper understanding of various imputation techniques and their suitability for different scenarios.
- Learned to differentiate between numerical and categorical variables and their subtypes.
- Enhanced skills in data exploration, preprocessing, and preparation for machine learning tasks.
- Improved understanding of how to use Pandas and Scikit-learn libraries for data manipulation and analysis.

Remember to replace `"your_dataset.csv"` with the actual file name and adapt the code to fit the specific characteristics of your chosen dataset. Good luck with your experiment!