MACHINE LEARNING

LAB WORK 1

Name: Arjun Unnikrishnan USN: 22BTRAD004

Boston Housing Dataset

Question 1. Load a dataset with missing values (Boston Housing Dataset).

Code:

import pandas as pd

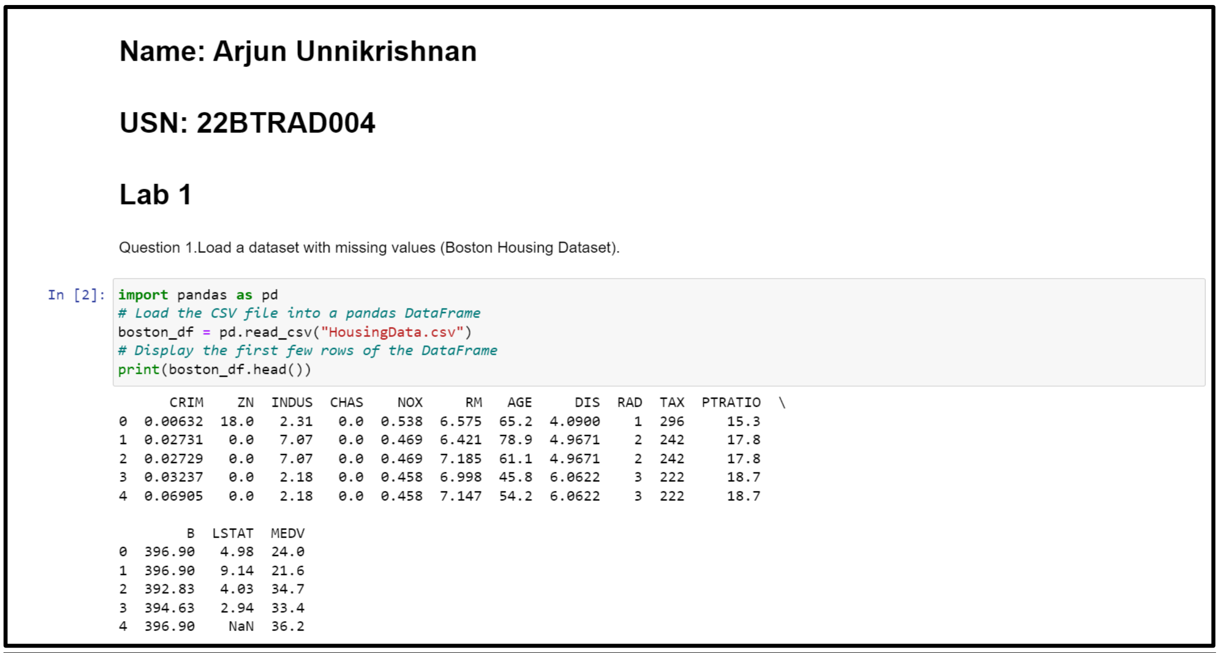
Load the CSV file into a pandas DataFrame

boston_df = pd.read_csv("HousingData.csv")

Display the first few rows of the DataFrame

print(boston_df.head())

Output:



Name: Arjun Unnikrishnan

USN: 22BTRAD004

Lab 1

Question 1.Load a dataset with missing values (Boston Housing Dataset).

```
In [2]: import pandas as pd
        # Load the CSV file into a pandas DataFrame
        boston_df = pd.read_csv("HousingData.csv")
        # Display the first few rows of the DataFrame
        print(boston_df.head())
```

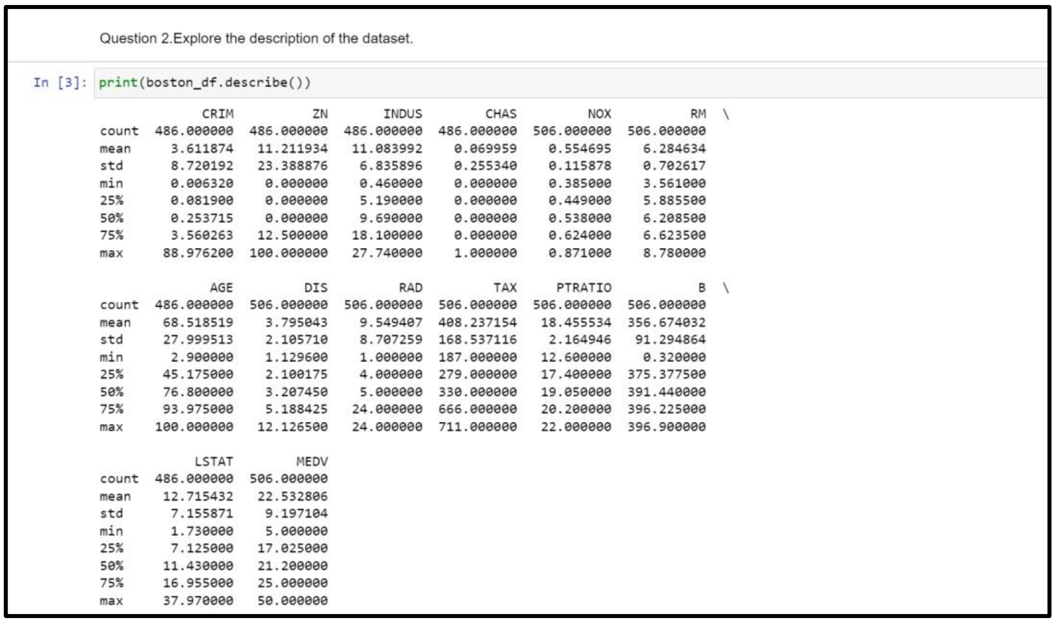
```
      CRIM    ZN  INDUS  CHAS    NOX     RM   AGE     DIS  RAD  TAX  PTRATIO  \
0  0.00632  18.0   2.31   0.0  0.538  6.575  65.2  4.0900    1  296     15.3
1  0.02731   0.0   7.07   0.0  0.469  6.421  78.9  4.9671    2  242     17.8
2  0.02729   0.0   7.07   0.0  0.469  7.185  61.1  4.9671    2  242     17.8
3  0.03237   0.0   2.18   0.0  0.458  6.998  45.8  6.0622    3  222     18.7
4  0.06905   0.0   2.18   0.0  0.458  7.147  54.2  6.0622    3  222     18.7

        B  LSTAT  MEDV
0  396.90   4.98  24.0
1  396.90   9.14  21.6
2  392.83   4.03  34.7
3  394.63   2.94  33.4
4  396.90    NaN  36.2
```

Question 2. Explore the description of the dataset.

Code:

print(boston_df.describe())

Output:



Question 2.Explore the description of the dataset.

```
In [3]: print(boston_df.describe())

             CRIM          ZN       INDUS        CHAS         NOX          RM  \
count  486.000000  486.000000  486.000000  486.000000  506.000000  506.000000
mean     3.611874   11.211934   11.083992    0.069959    0.554695    6.284634
std      8.720192   23.388876    6.835896    0.255340    0.115878    0.702617
min      0.006320    0.000000    0.460000    0.000000    0.385000    3.561000
25%      0.081900    0.000000    5.190000    0.000000    0.449000    5.885500
50%      0.253715    0.000000    9.690000    0.000000    0.538000    6.208500
75%      3.560263   12.500000   18.100000    0.000000    0.624000    6.623500
max     88.976200  100.000000   27.740000    1.000000    0.871000    8.780000

              AGE         DIS         RAD         TAX     PTRATIO           B  \
count  486.000000  506.000000  506.000000  506.000000  506.000000  506.000000
mean    68.518519    3.795043    9.549407  408.237154   18.455534  356.674032
std     27.999513    2.105710    8.707259  168.537116    2.164946   91.294864
min      2.900000    1.129600    1.000000  187.000000   12.600000    0.320000
25%     45.175000    2.100175    4.000000  279.000000   17.400000  375.377500
50%     76.800000    3.207450    5.000000  330.000000   19.050000  391.440000
75%     93.975000    5.188425   24.000000  666.000000   20.200000  396.225000
max    100.000000   12.126500   24.000000  711.000000   22.000000  396.900000

            LSTAT        MEDV
count  486.000000  506.000000
mean    12.715432   22.532806
std      7.155871    9.197104
min      1.730000    5.000000
25%      7.125000   17.025000
50%     11.430000   21.200000
75%     16.955000   25.000000
max     37.970000   50.000000
```

Question 3. Identify the number of missing values corresponding to each feature.

Code:

# Identify the number of missing values for each feature

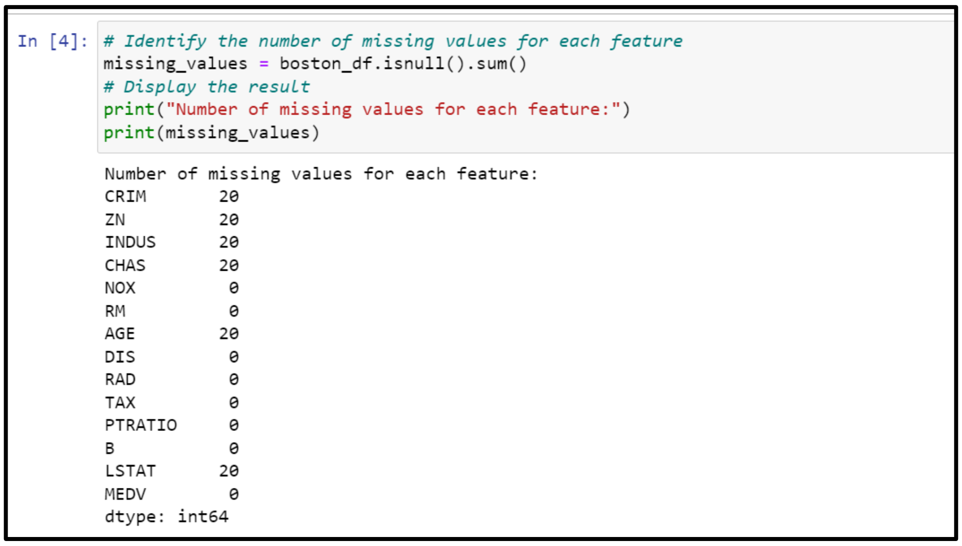
missing_values = boston_df.isnull().sum()

# Display the result

print("Number of missing values for each feature:")

print(missing_values)

Output:



```
In [4]: # Identify the number of missing values for each feature
        missing_values = boston_df.isnull().sum()
        # Display the result
        print("Number of missing values for each feature:")
        print(missing_values)

        Number of missing values for each feature:
        CRIM       20
        ZN         20
        INDUS      20
        CHAS       20
        NOX         0
        RM          0
        AGE        20
        DIS         0
        RAD         0
        TAX         0
        PTRATIO     0
        B           0
        LSTAT      20
        MEDV        0
        dtype: int64
```

Question 4. Explore and visualize the missing data patterns.

Code:

import missingno as msno

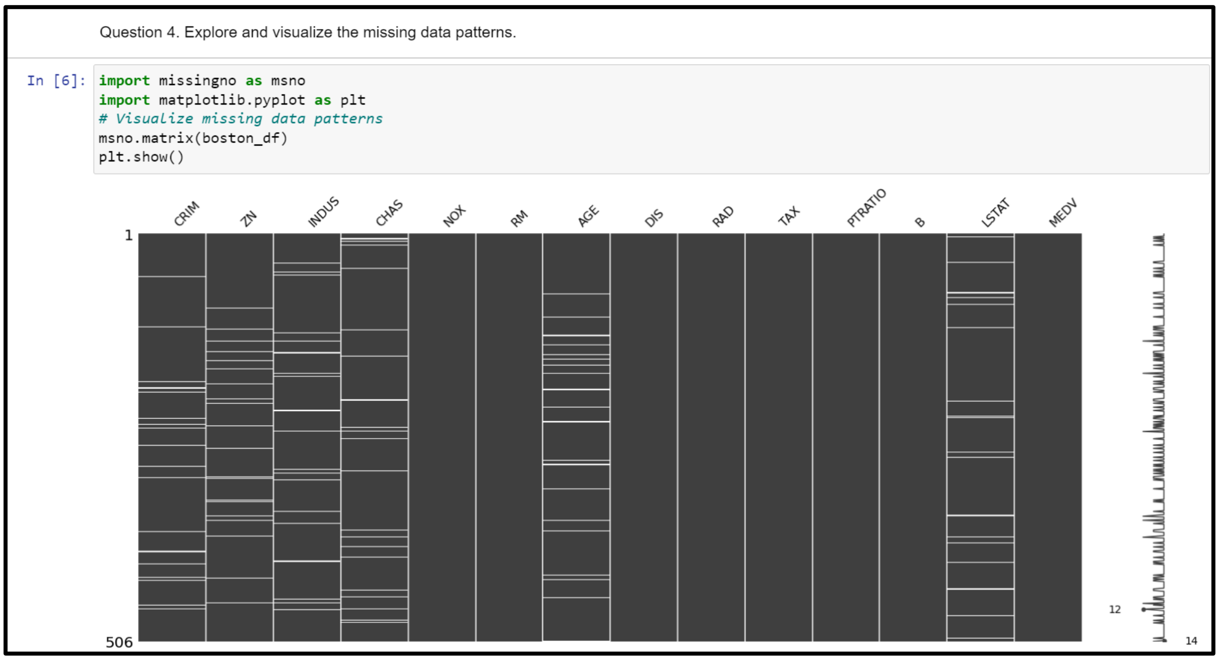
import matplotlib.pyplot as plt

Visualize missing data patterns

msno.matrix(boston_df)

plt.show()

Output:



Question 4. Explore and visualize the missing data patterns.

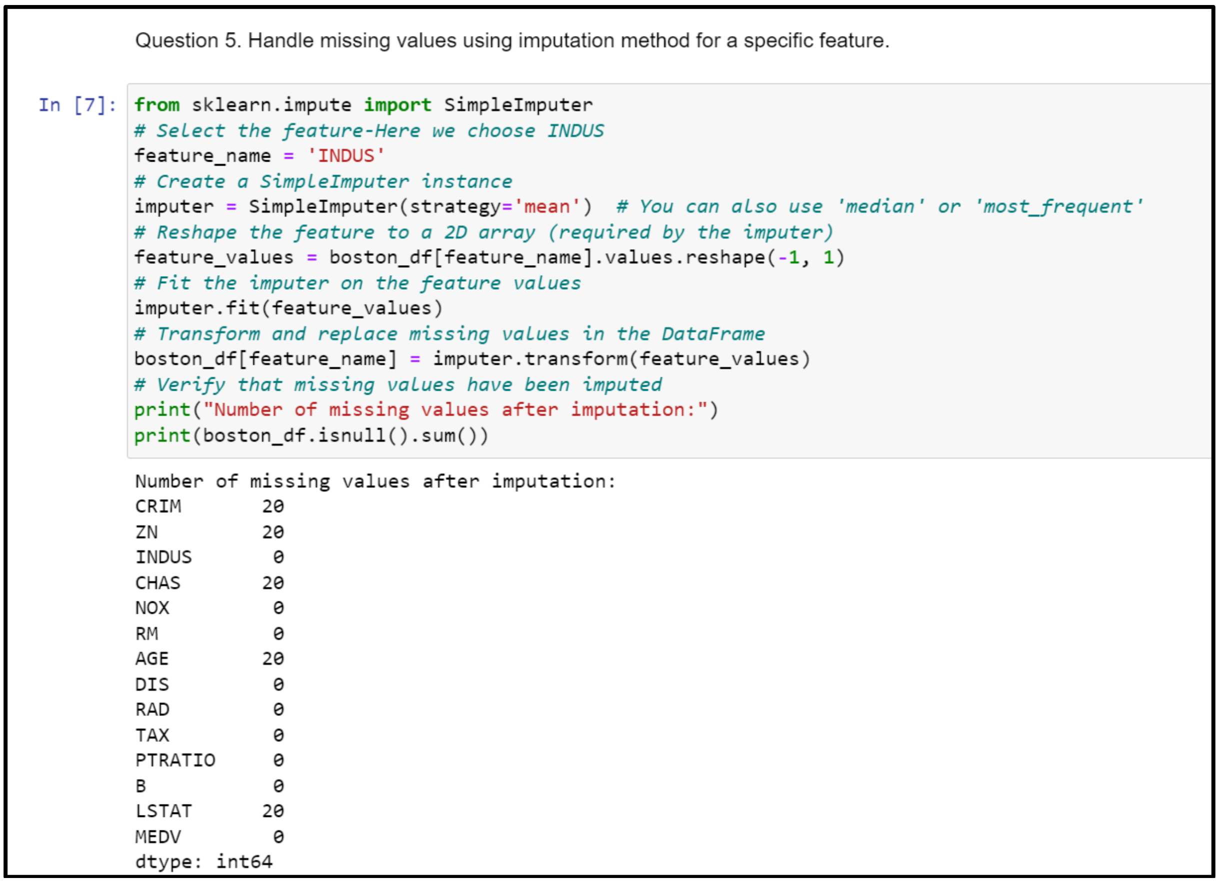
```
In [6]: import missingno as msno
        import matplotlib.pyplot as plt
        # Visualize missing data patterns
        msno.matrix(boston_df)
        plt.show()
```

Question 5. Handle missing values using imputation method for a specific feature.

Code:

```
from sklearn.impute import SimpleImputer
# Select the feature-Here we choose INDUS
feature_name = 'INDUS'
# Create a SimpleImputer instance
imputer = SimpleImputer(strategy='mean')  # You can also use 'median' or 'most_frequent'
# Reshape the feature to a 2D array (required by the imputer)
feature_values = boston_df[feature_name].values.reshape(-1, 1)
# Fit the imputer on the feature values
imputer.fit(feature_values)
# Transform and replace missing values in the DataFrame
boston_df[feature_name] = imputer.transform(feature_values)
# Verify that missing values have been imputed
print("Number of missing values after imputation:")
print(boston_df.isnull().sum())
```

Output:



Question 5. Handle missing values using imputation method for a specific feature.

```
In [7]: from sklearn.impute import SimpleImputer
        # Select the feature-Here we choose INDUS
        feature_name = 'INDUS'
        # Create a SimpleImputer instance
        imputer = SimpleImputer(strategy='mean')  # You can also use 'median' or 'most_frequent'
        # Reshape the feature to a 2D array (required by the imputer)
        feature_values = boston_df[feature_name].values.reshape(-1, 1)
        # Fit the imputer on the feature values
        imputer.fit(feature_values)
        # Transform and replace missing values in the DataFrame
        boston_df[feature_name] = imputer.transform(feature_values)
        # Verify that missing values have been imputed
        print("Number of missing values after imputation:")
        print(boston_df.isnull().sum())
```

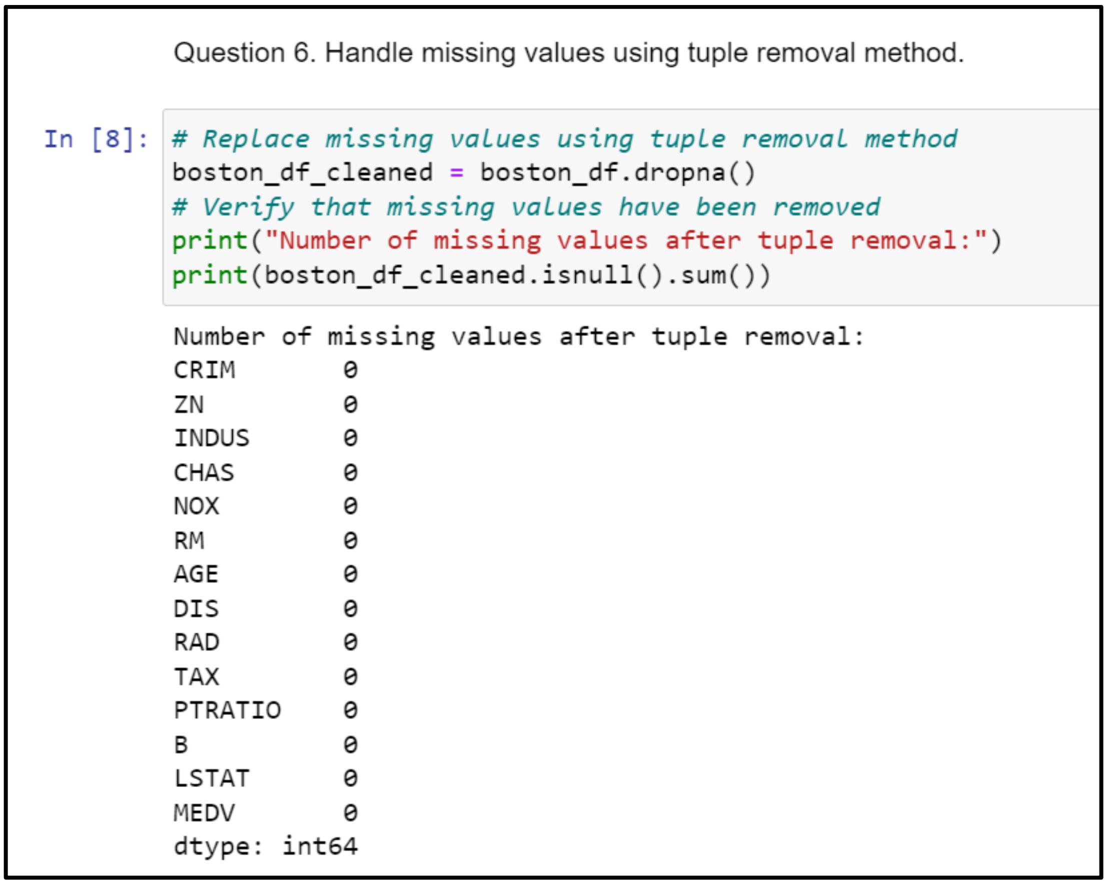
```
Number of missing values after imputation:
CRIM       20
ZN         20
INDUS       0
CHAS       20
NOX         0
RM          0
AGE        20
DIS         0
RAD         0
TAX         0
PTRATIO     0
B           0
LSTAT      20
MEDV        0
dtype: int64
```

Question 6. Handle missing values using tuple removal method.

Code:

```
# Replace missing values using tuple removal method
boston_df_cleaned = boston_df.dropna()
# Verify that missing values have been removed
print("Number of missing values after tuple removal:")
print(boston_df_cleaned.isnull().sum())
```

Output:



Question 6. Handle missing values using tuple removal method.

```
In [8]: # Replace missing values using tuple removal method
        boston_df_cleaned = boston_df.dropna()
        # Verify that missing values have been removed
        print("Number of missing values after tuple removal:")
        print(boston_df_cleaned.isnull().sum())
```

```
Number of missing values after tuple removal:
CRIM       0
ZN         0
INDUS      0
CHAS       0
NOX        0
RM         0
AGE        0
DIS        0
RAD        0
TAX        0
PTRATIO    0
B          0
LSTAT      0
MEDV       0
dtype: int64
```

GitHub Link: https://github.com/arj1-1n/ML