Amrita School of Computing

Department of Computer Science & Engineering

19CSE304 Foundations of Data Science

Lab Sheet #3 Data Preprocessing

Exercise 1 Generate Dataset

To generate a dataset in Python, you can use various libraries such as NumPy and Pandas. A sample code is below:

```
import numpy as np
import pandas as pd
# Set a random seed for reproducibility
np.random.seed(42)
# Generate random data for three features: Age, Income, and Score
num_samples = 100                    # Number of data points
age = np.random.randint(18, 65, size=num_samples)

income = np.random.normal(50000, 10000, size=num_samples)

score = np.random.uniform(0, 100, size=num_samples)

# Create a DataFrame to store the data
data = pd.DataFrame({
        'Age': age,
        'Income': income,
        'Score': score
})

# Print the first few rows of the dataset
print(data.head())
```

1. Generate a dataset with the Employee Information with the following features. Include some missing values too.

   Employee ID (unique identifier)
   Employee Name (text data)
   Department (categorical: 'Sales,' 'Marketing,' 'Engineering,' etc.)
   Salary (numerical)
   joining Date (date or datetime)

2. Find the Count/percentage of missing values in every column of the dataset.

Exercise II Missing Value Imputation

Dataset: titanic_dataset.csv (github.com) for Non Time Series problem

You can use 'missingno' Library which is a Python library used to visualize missing data in datasets. Visualizations using this library help you to understand and identify missing values in your data. To visualize missing data using missingno, you typically need a pandas DataFrame that contains your dataset.

Example:

```
import missingno as msno

import pandas as pd
import numpy as np

# Create a sample DataFrame with missing values
data = {
        'A': [1, 2, 3, np.nan, 5],
         'B': [6, np.nan, np.nan, 9, 10],
         'C': [11, 12, 13, 14, 15],
        }
df = pd.DataFrame(data)
# Create a matrix visualization of missing data
msno.matrix(df)
```

3. For the titanic dataset use Missingno library to visualize the missing values. You can use msno.bar(data), msno.matrix(data)
4. Create a copy of the dataframe. Drop rows which contain any Nan or missing value in fare column.
5. Replace the missing value in the column cabin with the most frequent value.
6. Replace missing values in the 'Embarked' column with the most common class.
7. **KNN Imputer:** K-Nearest Neighbors Imputer (KNN Imputer) is a technique for imputing or filling in missing values in a dataset using the K-Nearest Neighbors algorithm. It works by identifying the K nearest data points with complete information (i.e., non-missing values) for each data point with missing values and then imputing the missing values based on the values of those nearest neighbors.

```
import numpy as np

from sklearn.impute import KNNImputer

nan = np.nan

X = [[1, 2, nan], [3, 4, 3], [nan, 6, 5], [8, 8, 7]]

imputer = KNNImputer(n_neighbors=2, weights="uniform")

imputer.fit_transform(X)
```

Make use of the above KNNimputer to impute missing values in age column.

8. **Multivariate feature imputation:** Multivariate feature imputation also known as multivariate imputation, is a technique for imputing missing values in a dataset by taking into account relationships between multiple features (variables). Unlike univariate imputation, which considers each variable independently, multivariate imputation leverages the relationships between variables to make more informed imputations.

```
from sklearn.experimental import enable_iterative_imputer
from sklearn.impute import IterativeImputer
train_mice = train.copy(deep=True)
mice_imputer = IterativeImputer()
train_mice['Age'] = mice_imputer.fit_transform(train_mice[['Age']])
```

In the above code enable_iterative_imputer is used to enable the IterativeImputer,. A deep copy of the 'train' DataFrame is created, named 'train_mice'. A deep copy ensures that you're working with a new DataFrame that won't affect the original 'train' DataFrame. An instance of IterativeImputer is created as 'mice_imputer'. This imputer will be used to impute missing values in the 'Age' column.mice_imputer.fit_transform(train_mice[['Age']]) imputes missing values in the 'Age' column by considering the relationships with other features in the DataFrame. The resulting imputed values are assigned to the 'Age' column of the 'train_mice' DataFrame. This code updates the 'Age' column in 'train_mice' with imputed values based on the IterativeImputer's predictions. After running this code, the 'train_mice' DataFrame will have the missing values in the 'Age' column filled with imputed values.

By making use of multivariate feature imputation impute the missing values in the column body.

9. Imputation for Categorical Data
    a. Generate a categorical dataset as per below code

```
data = {"X1": [np.nan, "Red" , "Blue", "Red", np.nan,
      "Red", "Green", np.nan, "Blue", "Red"],
      "X2": ["Green", "Green", "Red", "Blue", "Green" ,
      "Blue" , np.nan, "Red", "Green", np.nan ]}
colors = pd.DataFrame(data)
print(colors)
```

    b. Imputation Method 1: Most Common Class

```
# for each column, get value counts in decreasing order and take the index (value) of most common class
df_most_common_imputed = colors.apply(lambda x:
x.fillna(x.value_counts().index[0]))
df_most_common_imputed
```

Imputation Method 2: "Unknown" Class
df_unknown_imputed = colors.fillna("Unknown")
df_unknown_imputed

c. Frequent Categorical Imputation[Mode Imputation]

```
for feature in missing_cf:
    print(df1[feature].mode())
```

Exercise III Remove Noise from Data

1. Load the dataset Cupcake.csv
2. Apply binning by distance: *Binning by distance is a technique used in data analysis and statistics to group data points into discrete bins or intervals based on their proximity or distance from a reference point.*
   For the cupcake dataset perform noise removal using binning by distance technique. For that, perform the following steps:
   a. Find the minimum and maximum values in the "Cupcake" column using the **min()** and **max()** functions
   b. Use the linspace() function of the numpy package to calculate the 4 bins, equally distributed.
   c. Define the labels as 'small', medium' and 'big'
   d. Use the Pandas **cut** function to convert the numeric values of the column Cupcake into the categorical values. Specify the bins and the labels. In addition, we set the parameter include_lowest to True in order to include also the minimum value.
   e. Plot the distribution of values, by using the hist() function of the matplotlib package.
3. Apply binning by frequency: *Binning by frequency calculates the size of each bin so that each bin contains the (almost) same number of observations, but the bin range will vary.*
   Perform binning by frequency on cupcake dataset.

   (Use the Python pandas qcut() function)

4. Sampling: *Sampling is another technique of data binning. It permits to reduce the number of samples, by grouping similar values or contiguous values. There are three approaches to perform sampling:*
   - *by bin means: each value in a bin is replaced by the mean value of the bin.*
   - *by bin median: each bin value is replaced by its bin median value.*
   - *by bin boundary: each bin value is replaced by the closest boundary value, i.e. maximum or minimum value of the bin.*
   - *In order to perform sampling, the binned_statistic() function of the scipy.stats package can be used. This function receives two arrays as input, x_data and y_data, as well as the statistics to be used (e.g. median or mean) and the number of bins to be created. The function returns the values of the bins as well as the edges of each bin.*
     a. Replace the bin value by the bin means.
     b. Replace the bin value by the bin median.
     c. Replace the bin value by the bin boundary.

   (Use the apply() function to apply the set_to_median() to the Cupcake column.)

Extra Credit Exercise (Optional):

Impute the missing values in the titanic dataset for the column age using KNN based imputer, without using default library function. Write an algorithm from scratch to implement k-nn and then use the predictions to impute the missing values.