Hands-On Exercise 1-1: Basic Data Exploration and Analysis

Instructor: Dr. Peng Kang

Section 1 - Data Types (1 point)

Objectives

The objective of this hands-on exercise is to deepen your understanding of various types of attributes that describe data objects. Through practical examples, you'll explore nominal, binary, ordinal, and numeric attributes, including their subtypes such as interval-scaled and ratio-scaled attributes. By the end of this exercise, you should be able to:

- Understand Python data structures that can represent different types of attributes.
- Create and manipulate datasets in Python.
- Visualize different attribute types using Python libraries.

Getting Started

ATTENTION: Please create a new notebook file for this part of exercise. Name this notebook "Hands-on Exercise 1-1-Section 1". Let's first set up our Python environment. We will use the Pandas library to manipulate data frames, and Matplotlib to visualize data.

```
import pandas as pd
import matplotlib.pyplot as plt
```

Data Initialization (0.2 points)

Now, let's initialize a simple data frame to work with. This data frame will represent a university database with attributes: student_ID, name, GPA, and classification.

```
# Initialize a dataframe
df = pd.DataFrame({
    'student_ID': [1, 2, 3, 4],
    'name': ['Alice', 'Bob', 'Catherine', 'David'],
    'major': ['CSC', 'EE', 'EDU', 'CSC'],
    'GPA': [3.2, 3.8, 2.5, 3.9],
    'classification': ['Junior', 'Senior', 'Sophomore', '
                                        Freshman']
})

df
```

Here, 'student_ID', 'name' and 'major' are nominal data, 'GPA' is ratio-scaled, and 'classification' is ordinal. For example, 'major' is a nominal attribute that can take values like 'CSC' for Computer Science, 'EE' for Electrical Engineering, etc. Similarly, 'classification' is an ordinal attribute, because the values ('Freshman', 'Sophomore', 'Junior', 'Senior') have a meaningful order.

Nominal Attributes (0.2 points)

Display Names

Display the 'name' column, which is a nominal attribute.

```
print(df['name'])
```

Observe that names are simply categories without any intrinsic order.

Explore Majors

Let's visualize the distribution of students across different majors.

```
# Plotting the distribution of Majors
df['major'].value_counts().plot(kind='bar')
plt.title('Distribution of Majors')
plt.xlabel('Major')
plt.ylabel('Count')
plt.show()
```

Binary Attributes (0.2 points)

0.1 Create Considering Graduate School Column

Let's add a column to our data frame that shows if a student is considering going to the graduate school.

```
df['considering_graduate_school'] = ['Yes', 'Yes', 'No', 'No']
df
```

0.2 Convert to Binary

Convert the graduate school column into a binary attribute.

```
df['considering_graduate_school_binary'] = df['
                                    considering_graduate_school'].
                                    map({'No': 0, 'Yes': 1})
df
```

Ordinal Attributes (0.2 points)

View Classification

View the 'classification' column.

```
print(df['classification'])
```

Order Classification

Convert 'classification' to ordinal numbers.

```
classification_map = {'Freshman': 1, 'Sophomore': 2, 'Junior': 3
                                    , 'Senior': 4}
df['classification_ordinal'] = df['classification'].map(
                                    classification_map)
df
```

Numeric Attributes (0.2 points)

View GPA

Display the 'GPA' column.

```
print(df['GPA'])
```

Section 2 - Basic Data Exploration (3 points)

Objectives

Upon completion of this hands-on exercise, you will be able to:

- Read data from Google Colab and Google Storage
- Perform basic data description and exploration
- Filter and group data
- Perform the Five Number Summary

Data Files Overview

In this hands-on experiment, you will work with two datasets:

- **geolocation.csv**: This file contains geolocation data for different truck drivers. Each row represents a single event, and the columns include truck ID, driver ID, event type (e.g., normal, overspeed), geolocation (latitude, longitude), and other attributes.
- **trucks.csv**: This file contains information about different trucks, their models, and fuel and mileage data for June and May 2013. The columns include driver ID, truck ID, model, mileage, and gas usage for the two months.

Getting Started

ATTENTION: Please create a new notebook file for this part of exercise. Name this notebook "Hands-on Exercise 1-1-Section 2".

Let's import the essential Python libraries and mount Google Drive to read our dataset. Google Drive can act as a rudimentary database for storing and reading data files.

```
from google.colab import drive
import pandas as pd
import matplotlib.pyplot as plt

# Mount Google Drive
drive.mount('/content/drive/')
```

Uploading Data Files to Google Drive

Before starting the experiment, make sure both `geolocation.csv` and `trucks.csv` are uploaded to your Google Drive. To upload files to Google Drive:

1. Open your Google Drive account in a web browser.
2. Navigate to the folder where you want to upload the files.
3. Click on the 'New' button at the top left corner.
4. Select 'File upload', then choose `geolocation.csv` and `trucks.csv` from your local machine to upload them.

Once the files are uploaded, take note of the folder paths as you will need them to read the files into Google Colab.

Reading the Data (0.5 points)

Next, let's read the `geolocation.csv` data file and load it into a Pandas DataFrame.

```
# Read the CSV file into a DataFrame
df = pd.read_csv('/content/drive/My Drive/path_to_file/
                                    geolocation.csv')

# Show the first few rows of the DataFrame
df.head()
```

Describing the Data (0.5 points)

To describe the data, we can use the `info()` and `describe()` methods from Pandas.

```
# Show data types and non-null counts
df.info()
```

```
# Five Number Summary
df.describe()
```

The `info()` and `describe()` methods in Pandas are incredibly useful for getting a quick overview of the dataset. Below, you'll find detailed information on what each method provides.

`info()` Method

When you run `df.info()`, the method will provide the following details about your DataFrame:

- **Range Index**: The number of entries in the DataFrame, starting and ending index values.
- **Data Columns**: The total number of columns in the DataFrame.
- **Column Names and Data Types**: For each column, you get its name along with the data type (e.g., int64, object, float64).
- **Non-Null Count**: For each column, the number of non-null entries is displayed. This is crucial for identifying columns that may have missing data.
- **Memory Usage**: Total memory usage for storing the DataFrame.

`describe()` Method

We will examine the `describe()` method and its outputs in much more details later in this exercise; however, when you run `df.describe()`, the method generates a table with the following statistical details for each numerical column:

- **count**: Number of non-null entries. Useful for identifying if there are any missing values.
- **mean**: The mean value of the column. This provides a sense of "central tendency" of the data.
- **std**: Standard deviation, which measures the amount of variation or dispersion in the data.
- **min**: Minimum value in the column (can be outliers).
- **25% (First Quantile)**: 25% of the data points are below this value.
- **50% (Median)**: The middle value when the data points are sorted. 50% of the data is below this value.

- **75% (Third Quantile)**: 75% of the data points are below this value.
- **max**: Maximum value in the column (can be outliers).

These statistical summaries give you a comprehensive overview of the distribution of data in each numerical column, helping to identify trends, anomalies, or possible areas for further investigation.

Filtering Unsafe Events

In the real world, focusing on normal or "safe" events often does not yield as much actionable insight as looking into the "unsafe" or outlier events. These are the cases that typically require immediate attention to prevent adverse outcomes. In the context of our `geolocation.csv` data, these unsafe events might indicate speeding, unsafe following distance, or other risky behaviors that warrant closer examination.

What is Filtering?

Filtering refers to the process of selecting a subset of your data based on certain conditions. In Pandas, this is often done using Boolean indexing, where each row is checked against a condition and only the rows that satisfy the condition are retained.

How to Filter Data in Pandas (0.5 points)

To filter out only the unsafe events, you would run:

```
# Filter rows where event is not normal
unsafe_df = df[df['event'] != 'normal']
unsafe_df.head()
```

Here's what's happening in this line of code:

- `df['event'] != 'normal'`: This part of the code checks each row in the DataFrame to see if the event is not equal to "normal". It returns a Boolean series (True or False).
- `df[...]`: The DataFrame is indexed by this Boolean series, so only the rows with "True" are retained.
- `unsafe_df`: This new DataFrame contains only the rows where the event was not "normal" (i.e., unsafe).

What to Expect After Filtering

After filtering, your new DataFrame (`unsafe_df`) will contain fewer rows than the original (`df`). Each row in this filtered DataFrame represents an "unsafe" event, providing a focused dataset for further analysis. You may want to check the dimensions of the new DataFrame using `unsafe_df.shape` and look at the first few rows using `unsafe_df.head()` to confirm that the filtering process was successful.

Next Steps

Once you have the filtered data, you can proceed to perform more targeted exploratory data analysis (EDA), statistical tests, or even predictive modeling specifically on these "unsafe" events.

Grouping Specific Events

We see there are lots of unsafe events in the dataset. So, let's try to count the number of rows of specific events.

```
# Group by event type and count the number of occurrences
grouped_df = unsafe_df.groupby('event').count()
grouped_df
```

Five Number Summary (0.5 points)

Refer to your Data Mining Book Chapter 2.2, pages 49-50 for a detailed explanation of the Five Number Summary. We can achieve this by using the `describe()` method.

```
# Five Number Summary for the 'velocity' column
df['velocity'].describe()
```

Importance of Five Number Summary

The Five Number Summary is fundamental in statistics and data science as it provides a quick snapshot of the distribution of a dataset. These five numbers include the minimum, first quartile (25th percentile), median (50th percentile), third quartile (75th percentile), and the maximum. Please note that in pandas, the minimum and maximum is the smallest and largest number of a specific attribute, thus they can be outliers, which is smaller than Q1 - 1.5IQR or larger than Q3 + 1.5IQR.

Individual Calculations

While there are functions to get all these values at once, such as the `describe()` method that we discussed earlier, understanding how to calculate them individually can provide a deeper understanding. Below are the methods to calculate these numbers separately for the *'velocity'* column:

- **Minimum:** The smallest number in the dataset.

```
min_value = df['velocity'].min()
min_value
```

- **First Quartile (Q1):** 25% of the data falls below this value. You can use the `quantile()` function with parameter 0.25.

```
Q1 = df['velocity'].quantile(0.25)
Q1
```

- **Median (Q2):** The middle value when the data is sorted. Median is known as the 50% percentile, or 2^{nd} quartile, or Q2, but usually simply referred to as the median.

```
median = df['velocity'].median()
median
```

- **Third Quartile (Q3):** 75% of the data falls below this value. You can use the `quantile()` function with parameter 0.75.

```
Q3 = df['velocity'].quantile(0.75)
Q3
```

- **Maximum:** The largest number in the dataset.

```
max_value = df['velocity'].max()
max_value
```

Using a Single Command

The Pandas `describe()` function can get you all these numbers at once, along with some other useful statistics.

```
five_num_summary = df['velocity'].describe()[['min', '25%', '50%
                                    ', '75%', 'max']]
five_num_summary
```

Understanding the Output

Here's a quick rundown on what each number can tell you about your data:

- **Minimum** and **Maximum** give you an idea of the range of your data.
- **Q1** and **Q3** can provide insights into the spread and skewness of your data.
- **Median** provides a measure of central tendency.

By examining these numbers, you can gain valuable insights into the distribution of your data, identify outliers, and make informed decisions for further data analysis tasks.

Additional Questions

Q1: Import trucks.csv and Create DataFrame (0.2 points)

Load the data, trucks.csv, into a DataFrame named `df_trucks`. Make sure to use CSV formatting and specify columns later.

```
# Read trucks.csv into df_trucks DataFrame
df_trucks = pd.read_csv('/content/drive/My Drive/path_to_file/
                                trucks.csv')
df_trucks
```

Q2: Show df_trucks (0.2 points)

Display only the first 5 rows of `df_trucks` and limit it to 7 columns.

```
# Display first 5 rows and selected columns
df_trucks = df_trucks[['driverid', 'truckid', 'model', '
                                jun13_miles', 'jun13_gas', '
                                may13_miles', 'may13_gas']]
df_trucks.head(5)
```

Q3: Count Specific Models (0.2 points)

Count the number of rows of specific truck models.

```
# Group by model and count
model_count = df_trucks.groupby('model').count()
model_count
```

Q4 & Q5: Five Number Summary for Mileage and Gas (0.4 points)

Show the '**Five Number Summary**' for mileage and gas usages for June and May 2013.

```
# Five Number Summary for mileage in June and May
df_trucks[['jun13_miles', 'may13_miles']].describe()
```

```
# Five Number Summary for gas in June and May
df_trucks[['jun13_gas', 'may13_gas']].describe()
```