Python assignment-1

Name: Prajwal Pavana Gunda

SRN:PES2UG22EC900

MCQ

1. Which of the following operators have lower precedence than “not” in Python?

a) +

b) and

c)==

d)|

ans: (b) and

2. Which of the following object does not support indexing?

a) tuple

b) list

c) dictionary

d)set

ans: (d)set

DESCRIPTIVE QUESTIONS

1. **Explain the concepts of joint, marginal, and conditional probability in the context of exploratory data analysis. Write the pandas code using pd.crosstab() to generate a two-way table for each type of probability for the 'Automatic' and 'FuelType' columns.**

Answer: In exploratory data analysis, two-way tables help us understand the relationship between two categorical variables. Probabilities derived from these tables give us deeper insights.

Joint Probability: This is the likelihood of two independent events occurring at the same time. In a two-way table, it represents the probability of a specific combination of categories from both variables. For example, the probability that a car is both 'Petrol' and 'Automatic'.

Marginal Probability: This is the probability of a single event occurring, irrespective of the outcome of the other variable. It is found in the 'All' row/column (the margins) of the table. For example, the overall probability that a car has an 'Automatic' gearbox.

Conditional Probability: This is the probability of an event occurring, given that another event has already occurred. For example, the probability that a car is 'Petrol',

given that we know it is 'Automatic'.

Python Code:

Python

```
import pandas as pd


# Assuming cars_data2 is the pre-loaded and cleaned DataFrame
# cars_data2 = pd.read_csv('Toyota.csv', index_col=0, na_values=["??","????"])
# cars_data2.dropna(inplace=True)


# Joint Probability (normalize over all values)
print("Joint Probability Table:")
[cite_start]joint_prob = pd.crosstab(index=cars_data2['Automatic'], columns=cars_data2['FuelType'], normalize=True, dropna=True) # [cite: 2702, 2704]
print(joint_prob)


# Marginal Probability (use margins=True)
print("\nMarginal Probability Table:")
[cite_start]marginal_prob = pd.crosstab(index=cars_data2['Automatic'], columns=cars_data2['FuelType'], normalize=True, margins=True, dropna=True) # [cite: 2714, 2720, 2724]
print(marginal_prob)


# Conditional Probability (given the gearbox type - normalize by row)
print("\nConditional Probability of FuelType given Automatic:")
```

[cite_start]cond_prob_index = pd.crosstab(index=cars_data2['Automatic'], columns=cars_data2['FuelType'], normalize='index', margins=True, dropna=True) # [cite: 2731, 2735, 2736, 2738]

print(cond_prob_index)

1. **Compare and contrast the Python sequence types: List, Tuple, and Set. Discuss their key differences in terms of mutability, ordering, uniqueness of elements, and syntax.**

Answer: Lists, Tuples, and Sets are all sequence data types in Python used to store collections of items, but they have distinct characteristics.

- **List**:

Syntax: Defined with square brackets, e.g., my list = [1, 'a', 2, 'a'].

  - **Mutability**: Lists are **mutable**, meaning their elements can be changed, added, or removed after creation (e.g., using append(), remove()).
  - **Ordering**: Lists are **ordered**, meaning the elements maintain the position in which they were added. They can be accessed via an integer index.

Uniqueness: Lists **allow duplicate** elements.

- **Tuple**:
  -

Syntax: Defined with parentheses, e.g., my tuple = (1, 'a', 2, 'a').

  - **Mutability**: Tuples are **immutable**. Once a tuple is created, its elements cannot be changed, added, or removed.

Ordering: Like lists, tuples are **ordered**, and their elements can be accessed via an index.

  - **Uniqueness**: Tuples also **allow duplicate** elements.

- **Set**:

Syntax: Defined with curly braces, e.g., my set = {1, 'a', 2}.

  - **Mutability**: Sets themselves are **mutable** (you can add or remove elements), but their elements must be immutable.
  - **Ordering**: Sets are **unordered**. The elements have no fixed position, and thus sets do not support indexing.

Uniqueness: Sets only store **unique** elements; any duplicates are automatically removed upon creation