Practical 3

Descriptive Statistics - Measures of Central Tendency and variability Perform the following operations on any open source dataset (e.g., data.csv)
1. Provide summary statistics (mean, median, minimum, maximum, standard deviation) for a dataset (age, income etc.) with numeric variables grouped by one of the qualitative (categorical) variable. For example, if your categorical variable is age groups and quantitative variable is income, then provide summary statistics of income grouped by the age groups. Create a list that contains a numeric value for each response to the categorical variable.
2. Write a Python program to display some basic statistical details like percentile, mean, standard deviation etc. of the species of 'Iris-setosa', 'Iris-versicolor' and 'Iris- versicolor' of iris.csv dataset. Provide the codes with outputs and explain everything that you do in this step.

In [ ]:
```
# 1. Import all required python libraries

import pandas as pd

import warnings
warnings.filterwarnings('ignore')
```

In [ ]:
```
#2. Load datset in g drive

df=pd.read_csv('F:/SWATI ENGG/2021-2022/DS and Big Data/PRACTICALS/Codes/Assign_3_Mall
print(df)
```

In [ ]:
```
#1. mount Google drive
#from google.colab import drive
#drive.mount("/content/gdrive")
#df=pd.read_csv('/content/gdrive/My Drive/Colab Notebooks/Datasets/Assign 3 Mall_Custo
```

Q1. Provide summary statistics (mean, median, minimum, maximum, standard deviation) for a dataset (age, income etc.) with numeric variables grouped by one of the qualitative (categorical) variable. For example, if your categorical variable is age groups and quantitative variable is income, then provide summary statistics of income grouped by the age groups. Create a list that contains a numeric value for each response to the categorical variable.

In [ ]:
```
df.shape
```

In [ ]:
```
df.info()
```

Two of the variables are categorical (labelled as 'object') while the remaining are numerical (labelled as 'int').

In [ ]:
```
df.isnull().sum()
```

In [ ]:
```
df1=df.copy()
```

In [ ]:
```
# delete CustomerID from data frame
df.drop(["CustomerID"], axis=1,inplace=True)
```

In [ ]:
```
df
```

In [ ]:
```
df.mean()
```

In [ ]:
```
#It is also possible to calculate the mean of a particular variable in a data, as show
# where we calculate the mean of the variables 'Age' and 'Income'.
print(df.loc[:,'Age'].mean())
```

In [ ]:
```
print(df.loc[:,'Annual Income (k$)'].mean())
```

In [ ]:
```
 the previous sections, we computed the column-wise mean. It is also possible to calcu
mean(axis = 1)[0:5]
```

Median In simple terms, median represents the 50th percentile, or the middle value of the data, that separates the distribution into two halves. The line of code below prints the median of the numerical variables in the data. The command df.median(axis = 0) will also give the same output.

In [ ]:
```
df.median()
```

From the output, we can infer that the median age of the applicants is 36 years, the median annual income is USD 61.5 k$, and the median of spending scores is 50 . If there is a difference between the mean and the median values of these variables, it is because of the distribution of the data.

It is also possible to calculate the median of a particular variable in a data.

In [ ]:
```
#to calculate a median of a particular column
print(df.loc[:,'Age'].median())
print(df.loc[:,'Annual Income (k$)'].median())
```

In [ ]:
```
#OR
df[["Age", "Annual Income (k$)"]].median()
```

Mode Mode represents the most frequent value of a variable in the data. This is the only central tendency measure that can be used with categorical variables, unlike the mean and the median which can be used only with quantitative data.

In [ ]:
```
df.mode()
```

In [ ]:
```
#OR
df.mode(axis = 0)
```

Measures of Dispersion In the previous sections, we have discussed the various measures of central tendency. However, as we have seen in the data, the values of these measures differ for many variables. This is because of the extent to which a distribution is stretched or squeezed. In statistics, this is measured by dispersion which is also referred to as variability, scatter, or spread. The most popular measures of dispersion are standard deviation, variance, and the interquartile range.

Standard Deviation Standard deviation is a measure that is used to quantify the amount of variation of a set of data values from its mean. A low standard deviation for a variable indicates that the data points tend to be close to its mean, and vice versa.

In [ ]:
```
df.std()
```

It is also possible to calculate the standard deviation of a particular variable, as shown in the first two lines of code below. The third line calculates the standard deviation for the first five rows.

In [ ]:
```
print(df.loc[:,'Age'].std())
print(df.loc[:,'Annual Income (k$)'].std())
```

In [ ]:
```
#calculate the standard deviation of the first five rows
df.std(axis = 1)[0:5]
```

Variance Variance is another measure of dispersion. It is the square of the standard deviation and the covariance of the random variable with itself. The line of code below prints the variance of all the numerical variables in the dataset. The interpretation of the variance is similar to that of the standard deviation.

In [ ]:
```
df.var()
```

Interquartile Range (IQR) The Interquartile Range (IQR) is a measure of statistical dispersion, and is calculated as the difference between the upper quartile (75th percentile) and the lower quartile (25th percentile). The IQR is also a very important measure for identifying outliers and could be visualized using a boxplot.

IQR can be calculated using the iqr() function. The first line of code below imports the 'iqr' function from the scipy.stats module, while the second line prints the IQR for the variable 'Age'.

In [ ]:
```
from scipy.stats import iqr
iqr(df['Age'])
```

Skewness:

Another useful statistic is skewness, which is the measure of the symmetry, or lack of it, for a real-valued random variable about its mean. The skewness value can be positive, negative, or undefined. In a perfectly symmetrical distribution, the mean, the median, and the mode will all have the same value. However, the variables in our data are not symmetrical, resulting in different values of the central tendency.

The skewness values can be interpreted in the following manner:

Highly skewed distribution: If the skewness value is less than −1 or greater than +1.

Moderately skewed distribution: If the skewness value is between −1 and −½ or between +½ and +1.

Approximately symmetric distribution: If the skewness value is between −½ and +½.

We can calculate the skewness of the numerical variables using the skew() function, as shown below.

In [ ]:
```
print(df.skew())
```

Putting Everything Together We have learned the measures of central tendency and dispersion, in the previous sections. It is important to analyse these individually, however, because there are certain useful functions in python that can be called upon to find these values. One such important function is the .describe() function that prints the summary statistic of the numerical variables.

In [ ]:
```
df.describe()
```

The above output prints the important summary statistics of all the numerical variables like the mean, median (50%), minimum, and maximum values, along with the standard deviation. We can also calculate the IQR using the 25th and 75th percentile values.

However, the 'describe()' function only prints the statistics for the quantitative or numerical variable. In order to print the similar statistics for all the variables, an additional argument, include='all', needs to be added, as shown in the line of code below.

In [ ]:
```
df.describe(include='all')
```

#Que: Provide summary statistics of income grouped by age group

In [ ]:
```
df2=df.groupby(['Genre', 'Age Group'])
df2.first()
```

In [ ]:
```
# grouping income as per the age groups and calculating its mean,median and mode value
df3=df.groupby(['Age Group', 'Annual Income (k$)']).mean()
df3
```

In [ ]:
```

df4=df.groupby(['Age Group', 'Annual Income (k$)']).median()
df4
```

In [ ]:
```
#how to find the mode using pandas groupby

df.groupby(['Age Group','Annual Income (k$)']).agg(lambda x:x.value_counts().index[0])


#value_counts().index[0] gives you the most frequent value in the Series based on the
#If you want to access the actual value associated with this index, you can use total_
```

In [ ]:
```
#List that contains a numeric value for each responce to the categorical value
df5=df.groupby(['Age Group', 'Age', 'Annual Income (k$)'])
df5.first()
```

2. Write a Python program to display some basic statistical details like percentile, mean, standard deviation etc. of the species of 'Iris-setosa', 'Iris-versicolor' and 'Iris- versicolor' of iris.csv dataset.

In [ ]:
```
import pandas as pd
iris_df = pd.read_csv('F:/SWATI ENGG/2021-2022/DS and Big Data/PRACTICALS/Codes/Iris.c
```

In [ ]:
```python
# Read the iris dataset from CSV

# Calculate statistics function
def calculate_statistics(data):
    return {
        "Mean": data.mean(),
        "Standard Deviation": data.std(),
        "Median": data.median(),
        "Minimum": data.min(),
        "Maximum": data.max(),
        "25th Percentile": data.quantile(0.25),
        "50th Percentile": data.quantile(0.50),
        "75th Percentile": data.quantile(0.75)
    }

# Calculate statistics for each species
for species in iris_df['Species'].unique():
    print(f"\nStatistics for {species}:")
    species_data = iris_df[iris_df['Species'] == species].iloc[:, :-1]
    species_stats = calculate_statistics(species_data)
    for key, value in species_stats.items():
        print(f"{key}: {value}")
```

In [ ]:
```python
#write the above code without using inbuilt functions for mean, median, mode, percenti
```