pratical-3

April 17, 2024

1 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.

[ ]:

```
# # Import required libraries
# import pandas as pd
# import numpy as np
# import sklearn
# from sklearn import datasets
# from sklearn import preprocessing
# from sklearn.preprocessing import OneHotEncoder
# import matplotlib.pyplot as plt
# import seaborn as sns
# from IPython.display import display

# df1 = pd.read_csv('Customers.csv')
# df1

# col = ['CustomerID','Age','Annual Income (k$)', 'Spending Score (1-100)']
# # To find mean of all columns
# display(df1[col].mean())

# # To find mean of specific columns
# df1.loc[:,'Age'].mean()
# # To find mean row wise
# df1[col].mean(axis=1)[0:4]

# # To find median of all columns
# df1[col].median()

# # To find median of specific columns
# df1.loc[:,'Age'].median()

# # To find median row wise
# df1[col].median(axis=1)[0:4]

# # To find mode of all columns
# df1.mode()

# # To find mode of specific columns
# df1.loc[:,'Age'].mode()

# # To find minimun value of all columns
# df1.min()

# # To find minimum value of specific columns
# df1.loc[:,'Age'].min(skipna = False)

# # To find maximum value of all columns
# df1.max()

# # To find maximum value of specific column
# df1.loc[:,'Age'].max(skipna = False)

# # To find standard devitation of all columns
# df1[col].std()

# # To find standard devitation of specific column
# df1.loc[:,'Age'].std()

# # To find standard devitation row wise
# df1[col].std(axis=1)[0:4]

# # Categorical Variable: Genre
# # Quantitative Variable: Age
# df1.groupby(['Gender'])['Age'].mean()

# # Categorical Variable: Genre
# # Quantitative Variable: Income
# df_u = df1.rename(columns = {'Annual Income (k$)':'Income'},inplace = False)
# (df_u.groupby(['Gender']).Income.mean())
# # To create a list that contains a numeric value for each response to the␣
↪categorical variable.
# enc = preprocessing.OneHotEncoder()
# enc_df = pd.DataFrame(enc.fit_transform(df1[['Gender']]).toarray())
# enc_df

# # To concat numerical list to dataframe
# df_encode = df_u.join(enc_df)
# df_encode

# csv_url = 'https://archive.ics.uci.edu/ml/machine-learning-databases/iris/
↪iris.data'
# # Assign Column names
# col_names =␣
↪['Sepal_Length','Sepal_Width','Petal_Length','Petal_Width','Species']
# # Load Iris.csv into a Pandas data frame
# iris = pd.read_csv(csv_url, names = col_names)

# # Load all rows with Iris-setosa species in variable irisSet
# irisSet = (iris['Species']== 'Iris-setosa')
# # To display basic statistical details like percentile, mean,standard␣
↪deviation
# print('Iris-setosa')
# # For Iris-setosa use describe
# print(iris[irisSet].describe())

# # Load all rows with Iris-versicolor species in variable irisSet
# irisVer = (iris['Species']== 'Iris-versicolor')
# # To display basic statistical details like percentile, mean,standard␣
↪deviation
# print('Iris-versicolor')
# # For Iris-versicolor use describe
# print(iris[irisVer].describe())

# # Load all rows with Iris-virginica species in variable irisSet
# irisVir = (iris['Species']== 'Iris-virginica')
# # To display basic statistical details like percentile, mean,standard␣
↪deviation
# print('Iris-virginica')
# # For Iris-virginica use describe
# print(iris[irisVir].describe())
```

[1]:

```
import pandas as pd
import numpy as np
import sklearn
```

```
from sklearn import datasets
from sklearn import preprocessing
from sklearn.preprocessing import OneHotEncoder
import matplotlib.pyplot as plt
import seaborn as sns
from IPython.display import display
```

```
[2]: df1 = pd.read_csv('Customers.csv')
```

```
[3]: df1
```

```
[3]:      CustomerID  Gender  Age  Annual Income (k$)  Spending Score (1-100)
     0             1    Male   19                  15                      39
     1             2    Male   21                  35                      81
     2             3  Female   20                  86                       6
     3             4  Female   23                  59                      77
     4             5  Female   31                  38                      40
     ..          …     …   …                 …                     …
     195         196  Female   35                  46                      79
     196         197  Female   45                  78                      28
     197         198    Male   32                   4                      74
     198         199    Male   32                  81                      18
     199         200    Male   30                  33                      83

     [200 rows x 5 columns]
```

```
[4]: col = ['CustomerID','Age','Annual Income (k$)', 'Spending Score (1-100)']

     # To find mean of all columns
     display(df1[col].mean())
```

```
CustomerID                100.50
Age                        38.85
Annual Income (k$)         54.18
Spending Score (1-100)     50.20
dtype: float64
```

```
[5]: # To find mean of specific columns
     df1.loc[:,'Age'].mean()
```

```
[5]: 38.85
```

```
[6]: # To find mean row wise
     df1[col].mean(axis=1)[0:4]
```

```
[6]: 0    18.50
     1    34.75
```

```
2    28.75
3    40.75
dtype: float64
```

```
[7]: # To find median of all columns
     df1[col].median()
```

```
[7]: CustomerID                100.5
     Age                        36.0
     Annual Income (k$)         56.5
     Spending Score (1-100)     50.0
     dtype: float64
```

```
[8]: # To find median of specific columns
     df1.loc[:,'Age'].median()
```

```
[8]: 36.0
```

```
[9]: # To find median row wise
     df1[col].median(axis=1)[0:4]
```

```
[9]: 0    17.0
     1    28.0
     2    13.0
     3    41.0
     dtype: float64
```

```
[10]: # To find mode of all columns
      df1.mode()
```

```
[10]:      CustomerID  Gender   Age  Annual Income (k$)  Spending Score (1-100)
      0             1  Female  32.0                97.0                    42.0
      1             2     NaN   NaN                 NaN                     NaN
      2             3     NaN   NaN                 NaN                     NaN
      3             4     NaN   NaN                 NaN                     NaN
      4             5     NaN   NaN                 NaN                     NaN
      ..          …       …    …                  …                      …
      195         196     NaN   NaN                 NaN                     NaN
      196         197     NaN   NaN                 NaN                     NaN
      197         198     NaN   NaN                 NaN                     NaN
      198         199     NaN   NaN                 NaN                     NaN
      199         200     NaN   NaN                 NaN                     NaN

      [200 rows x 5 columns]
```

```
[11]: # To find mode of specific columns
      df1.loc[:,'Age'].mode()
```

```
[11]: 0    32
      Name: Age, dtype: int64
```

```
[12]: # To find minimun value of all columns
      df1.min()
```

```
[12]: CustomerID                       1
      Gender                      Female
      Age                             18
      Annual Income (k$)               0
      Spending Score (1-100)           1
      dtype: object
```

```
[12]: # To find minimum value of specific columns
      df1.loc[:,'Age'].min(skipna = False)
```

```
[12]: 18
```

```
[13]: # To find maximum value of all columns
      df1.max()
```

```
[13]: CustomerID                   200
      Gender                      Male
      Age                           70
      Annual Income (k$)           120
      Spending Score (1-100)        99
      dtype: object
```

```
[14]: # To find maximum value of specific column
      df1.loc[:,'Age'].max(skipna = False)
```

```
[14]: 70
```

```
[15]: # To find standard devitation of all columns
      df1[col].std()
```

```
[15]: CustomerID                57.879185
      Age                       13.969007
      Annual Income (k$)        29.865487
      Spending Score (1-100)    25.823522
      dtype: float64
```

```
[16]: # To find standard devitation of specific column
      df1.loc[:,'Age'].std()
```

```
[16]: 13.969007331558883
```

[17]:

```
# To find standard devitation row wise
df1[col].std(axis=1)[0:4]
```

[17]:

```
0    15.695010
1    33.668729
2    38.879086
3    33.230257
dtype: float64
```

[18]:

```
# Categorical Variable: Genre
# Quantitative Variable: Age

df1.groupby(['Gender'])['Age'].mean()
```

[18]:

```
Gender
Female    38.098214
Male      39.806818
Name: Age, dtype: float64
```

[19]:

```
# Categorical Variable: Genre
# Quantitative Variable: Income

df_u = df1.rename(columns = {'Annual Income (k$)':'Income'},inplace = False)
(df_u.groupby(['Gender']).Income.mean())
```

[19]:

```
Gender
Female    53.955357
Male      54.465909
Name: Income, dtype: float64
```

[20]:

```
# To create a list that contains a numeric value for each response to the␣
 ↪categorical variable.

enc = preprocessing.OneHotEncoder()
enc_df = pd.DataFrame(enc.fit_transform(df1[['Gender']]).toarray())
enc_df
```

[20]:

```
       0    1
0    0.0  1.0
1    0.0  1.0
2    1.0  0.0
3    1.0  0.0
4    1.0  0.0
..   …    …
195  1.0  0.0
196  1.0  0.0
197  0.0  1.0
```

```
198  0.0  1.0
199  0.0  1.0

[200 rows x 2 columns]
```

[21]:
```
# To concat numerical list to dataframe
df_encode = df_u.join(enc_df)
```

[22]:
```
df_encode
```

[22]:
```
     CustomerID  Gender  Age  Income  Spending Score (1-100)    0    1
0             1    Male   19      15                      39  0.0  1.0
1             2    Male   21      35                      81  0.0  1.0
2             3  Female   20      86                       6  1.0  0.0
3             4  Female   23      59                      77  1.0  0.0
4             5  Female   31      38                      40  1.0  0.0
..          …      …   …     …                    …   …   …
195         196  Female   35      46                      79  1.0  0.0
196         197  Female   45      78                      28  1.0  0.0
197         198    Male   32       4                      74  0.0  1.0
198         199    Male   32      81                      18  0.0  1.0
199         200    Male   30      33                      83  0.0  1.0

[200 rows x 7 columns]
```

1.0.1 Display basic statistical details on the iris dataset

Algorithm:

[23]:
```
csv_url = 'https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.
  ↪data'
```

[24]:
```
# Assign Column names
col_names =␣
  ↪['Sepal_Length','Sepal_Width','Petal_Length','Petal_Width','Species']
```

[25]:
```
# Load Iris.csv into a Pandas data frame
iris = pd.read_csv(csv_url, names = col_names)
```

[26]:
```
# Load all rows with Iris-setosa species in variable irisSet
irisSet = (iris['Species']== 'Iris-setosa')
# To display basic statistical details like percentile, mean,standard deviation
print('Iris-setosa')
# For Iris-setosa use describe
print(iris[irisSet].describe())



# Load all rows with Iris-versicolor species in variable irisSet
```

```
irisVer = (iris['Species']== 'Iris-versicolor')
# To display basic statistical details like percentile, mean,standard deviation
print('Iris-versicolor')
# For Iris-versicolor use describe
print(iris[irisVer].describe())

# Load all rows with Iris-virginica species in variable irisSet
irisVir = (iris['Species']== 'Iris-virginica')
# To display basic statistical details like percentile, mean,standard deviation
print('Iris-virginica')
# For Iris-virginica use describe
print(iris[irisVir].describe())
```

```
Iris-setosa
       Sepal_Length  Sepal_Width  Petal_Length  Petal_Width
count      50.00000    50.000000     50.000000     50.00000
mean        5.00600     3.418000      1.464000      0.24400
std         0.35249     0.381024      0.173511      0.10721
min         4.30000     2.300000      1.000000      0.10000
25%         4.80000     3.125000      1.400000      0.20000
50%         5.00000     3.400000      1.500000      0.20000
75%         5.20000     3.675000      1.575000      0.30000
max         5.80000     4.400000      1.900000      0.60000
Iris-versicolor
       Sepal_Length  Sepal_Width  Petal_Length  Petal_Width
count     50.000000    50.000000     50.000000    50.000000
mean       5.936000     2.770000      4.260000     1.326000
std        0.516171     0.313798      0.469911     0.197753
min        4.900000     2.000000      3.000000     1.000000
25%        5.600000     2.525000      4.000000     1.200000
50%        5.900000     2.800000      4.350000     1.300000
75%        6.300000     3.000000      4.600000     1.500000
max        7.000000     3.400000      5.100000     1.800000
Iris-virginica
       Sepal_Length  Sepal_Width  Petal_Length  Petal_Width
count      50.00000    50.000000     50.000000     50.00000
mean        6.58800     2.974000      5.552000      2.02600
std         0.63588     0.322497      0.551895      0.27465
min         4.90000     2.200000      4.500000      1.40000
25%         6.22500     2.800000      5.100000      1.80000
50%         6.50000     3.000000      5.550000      2.00000
75%         6.90000     3.175000      5.875000      2.30000
max         7.90000     3.800000      6.900000      2.50000
```

[ ]: