TC2-DS- Experiment 1

- **SIA VASHIST**
- PRN: 20190802107

AIM - Write a python program to open Comma Separated Value (CSV) and perform given statistical operations.

- **THEORY**:

> **What is CSV**?

> A CSV file, often known as a comma separated values file, enables the saving of data in plain text in a table-structured format. Every line in the file represents a data record, and each record is made up of one or more fields that are separated by commas. CSVs resemble spreadsheets but have a ".csv" extension.

> They differ from other spreadsheet file types in that you can only have a single sheet in a file, that cell, column, or row formatting cannot be saved, and that formulae cannot be saved.

Data is the value you record in your data sheet and is a specific measurement of a variable.

Data typically falls into one of two categories:

1. **Categorical data** represents groupings/classifications.
2. **Numerical/Quantitative data** represents quantities/amounts.

Variables with quantitative data are considered to be **quantitative variables**, while variables with categorical data are considered to be **categorical variables**. Each of these variable kinds can be further subdivided into other types.

- **Categorical Variables** :

It represent groupings of some kind. They are sometimes recorded as numbers, but the numbers represent categories rather than actual amounts of things.

> There are three types of categorical variables: binary, nominal, and ordinal variables.

> - **Binary Variable**: A binary variable is a categorical variable that can only have one of two possible values. These variables are typically expressed as Boolean variables (True or False) or integer variables (0 or 1). Example: Heads/tails in a coin flip

> - **Nominal Variable**: A nominal variable is a categorical variable that groups with no rank or order between them. Example: Colors.

> - **Ordinal Variable**: An ordinal variable is a categorical variable that ranks groups in a particular order. Example: Finishing place in a race.

- **Numerical Variables** :

The numbers you record when collecting quantitative data represent real amounts that can be added, subtracted, divided, and so on.

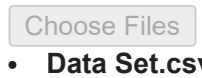
> There are two types of numerical variables: discrete, continuous variables.

> - **Discrete Variable**: A variable is considered discrete if the data it represents is a count of individual items or values. Example: Number of students in a class.

> - **Continuous Variable**: A variable is considered continuous if the data it represents is continuous or non-finite. Example: Age, Distance.

```
import collections
import pandas as pd
import io
from sklearn.preprocessing import OneHotEncoder, LabelEncoder, OrdinalEncoder
import seaborn as sns
import matplotlib.pyplot as plt


from google.colab import files
uploaded = files.upload()
```



- **Data Set.csv**(text/csv) - 5107 bytes, last modified: 9/20/2022 - 100% done

```
Saving Data Set.csv to Data Set (1).csv
```

```
Dataset_df = pd.read_csv(io.BytesIO(uploaded["Data Set.csv"]))
print("The Dataset is as Follows:")
print(Dataset_df.dropna(), '\n')
```

```
The Dataset is as Follows:
      Id  SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm  \
0      1            5.1           3.5            1.4           0.2
1      2            4.9           3.0            1.4           0.2
2      3            4.7           3.2            1.3           0.2
3      4            4.6           3.1            1.5           0.2
4      5            5.0           3.6            1.4           0.2
..   ...            ...           ...            ...           ...
145  146            6.7           3.0            5.2           2.3
146  147            6.3           2.5            5.0           1.9
147  148            6.5           3.0            5.2           2.0
148  149            6.2           3.4            5.4           2.3
149  150            5.9           3.0            5.1           1.8

            Species
0       Iris-setosa
1       Iris-setosa
2       Iris-setosa
3       Iris-setosa
4       Iris-setosa
..              ...
145  Iris-virginica
146  Iris-virginica
147  Iris-virginica
148  Iris-virginica
149  Iris-virginica

[150 rows x 6 columns]
```

Classifications of data:

```
print("The Classifications of Data are:")
print(Dataset_df.dtypes, '\n')
```

```
The Classifications of Data are:
Id                int64
SepalLengthCm    float64
SepalWidthCm     float64
PetalLengthCm    float64
PetalWidthCm     float64
Species           object
dtype: object
```

```
Categorical_columns = Dataset_df.dtypes[Dataset_df.dtypes == 'object'].index
print('Number of categorical variables:', len(Categorical_columns), '\n')
print(Categorical_columns)
```

```
Number of categorical variables: 1

Index(['Species'], dtype='object')
```

```
Numerical_columns = Dataset_df.dtypes[Dataset_df.dtypes != 'object'].index
print('Number of numerical variables:', len(Numerical_columns), '\n')
print(Numerical_columns)
```

```
Number of numerical variables: 5

Index(['Id', 'SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm', 'PetalWidthCm'], dtyp
```

Contingency of the table:

```
print("Contingency of the table:")
print(pd.crosstab(Dataset_df.Species, Dataset_df.SepalLengthCm, margins=True), '\n')
```

```
Contingency of the table:
SepalLengthCm    4.3  4.4  4.5  4.6  4.7  4.8  4.9  5.0  5.1  5.2  ...  6.9  \
Species                                                             ...
Iris-setosa        1    3    1    4    2    5    4    8    8    3  ...    0
Iris-versicolor    0    0    0    0    0    0    1    2    1    1  ...    1
Iris-virginica     0    0    0    0    0    0    1    0    0    0  ...    3
All                1    3    1    4    2    5    6   10    9    4  ...    4

SepalLengthCm    7.0  7.1  7.2  7.3  7.4  7.6  7.7  7.9  All
Species
Iris-setosa        0    0    0    0    0    0    0    0   50
Iris-versicolor    1    0    0    0    0    0    0    0   50
Iris-virginica     0    1    3    1    1    1    4    1   50
All                1    1    3    1    1    1    4    1  150

[4 rows x 36 columns]
```

Statistical Operations/Calculations:

1. Mean:

```
# Function that Calculates Mean:
def mean(n):
    return sum(n) / len(n)

# Function that Calculates Median:
def median(n):
    length = len(n)
    if length % 2 == 0:
        median1 = n[length // 2]
        median2 = n[length // 2 - 1]
        median = (median1 + median2) / 2
    else:
```

```
        median = n[length // 2]
    return median


# Function that Calculates Mode:
def mode(n):
    length = len(n)
    data = collections.Counter(n)
    d_list = dict(data)
    max_value = max(list(data.values()))
    mode_val = [num for num, freq in d_list.items() if freq == max_value]
    if len(mode_val) == length:
        return
    else:
        return ', '.join(map(str, mode_val))


# Function that Calculates Variance:
def variance(n):
    ans = sum((x - mean(n)) ** 2 for x in n) / len(n)
    return ans


# Function that Calculates Standard Deviation:
def standard_deviation(n):
    Stan_Deviation = variance(n) ** 0.5
    return Stan_Deviation


# Function that Calculates Quartile Range:
def inter_quartile_range(arr):
    arr = sorted(arr, reverse=False)
    if len(arr) % 2 != 0:
        mid_i = int(len(arr) / 2)
        arr_1 = arr[:mid_i]
        arr_2 = arr[mid_i + 1:]
        arr_1_mid = arr_1[int(len(arr_1) / 2)]
        arr_2_mid = arr_2[int(len(arr_2) / 2)]
        return arr_2_mid - arr_1_mid
    else:
        first_i = int(len(arr) / 2)
        second_i = int(len(arr) / 2)
        arr_1 = arr[:first_i]
        arr_1_mid_1 = arr_1[int(len(arr_1) / 2) - 1]
        arr_1_mid_2 = arr_1[int(len(arr_1) / 2)]
        arr_1_mid = sum([arr_1_mid_1, arr_1_mid_2]) / 2
        arr_2 = arr[second_i:]
        arr_2_mid_1 = arr_2[int(len(arr_2) / 2) - 1]
        arr_2_mid_2 = arr_2[int(len(arr_2) / 2)]
        arr_2_mid = sum([arr_2_mid_1, arr_2_mid_2]) / 2
        return arr_2_mid - arr_1_mid


print("Mean is: ", mean(Dataset_df["SepalLengthCm"]))
print("Median is: ", median(Dataset_df["SepalLengthCm"]))
```

```
print("Mode is: ", mode(Dataset_df["SepalLengthCm"]))
print("Variance is: ", variance(Dataset_df["SepalLengthCm"]))
print("Standard Deviation is: ", standard_deviation(Dataset_df["SepalLengthCm"]))
print("Quartile Range is: ", inter_quartile_range(Dataset_df["SepalLengthCm"]))
```

```
Mean is:  5.843333333333335
Median is:  6.5
Mode is:  5.0
Variance is:  0.6811222222222222
Standard Deviation is:  0.8253012917851409
Quartile Range is:  1.3000000000000007
```

Determination of type of categorical variable:

```
#Function that Calculates Nominal:
def Nominal(n):
    if len(set(n)) > len(n) // 5:
        return
    elif len(set(n)) == len(n):
        print("The Data is Nominal.")
    elif len(set(n)) <= 10:
        print("The Data is Nominal.")


def binary(n):
    if len(set(n)) == 2:
        print("The Data is Binary. ")
    else:
        return


def Ordinal(n):
    if len(set(n)) >= len(n) // 5:
        print("The Data is Ordinal.")
    else:
        return


def Category(n):
    binary(n)
    Nominal(n)
    Ordinal(n)


Category(Dataset_df["Species"])
```

```
The Data is Nominal.
```

Converting to Nominal data

```
print("The Nominal data is given as: ")

data_nominal = Dataset_df.copy()
lb_make = LabelEncoder()
data_nominal = lb_make.fit_transform(data_nominal["Species"])
print(data_nominal)  # Results in appending a new column to df
```

```
The Nominal data is given as:
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2]
```

Converting to Ordinal data

```
print("Ordinal Data is given as: ")
data_ordinal = Dataset_df.copy()
# define ordinal encoding
encoder2 = OrdinalEncoder()
# transform data
data_ordinal = encoder2.fit_transform(data_ordinal)
print(data_ordinal)
```

```
[ 93.   7.   2.  10.   6.   1.]
[ 94.  13.   6.  18.   9.   1.]
[ 95.  14.   9.  18.   8.   1.]
[ 96.  14.   8.  18.   9.   1.]
[ 97.  19.   8.  19.   9.   1.]
[ 98.   8.   4.   9.   7.   1.]
[ 99.  14.   7.  17.   9.   1.]
[100.  20.  12.  36.  21.   2.]
[101.  15.   6.  27.  15.   2.]
[102.  28.   9.  35.  17.   2.]
[103.  20.   8.  32.  14.   2.]
[104.  22.   9.  34.  18.   2.]
[105.  32.   9.  40.  17.   2.]
[106.   6.   4.  21.  13.   2.]
[107.  30.   8.  38.  14.   2.]
[108.  24.   4.  34.  14.   2.]
[109.  29.  15.  37.  21.   2.]
[110.  22.  11.  27.  16.   2.]
[111.  21.   6.  29.  15.   2.]
[112.  25.   9.  31.  17.   2.]
[113.  14.   4.  26.  16.   2.]
[114.  15.   7.  27.  20.   2.]
[115.  21.  11.  29.  19.   2.]
[116.  22.   9.  31.  14.   2.]
[117.  33.  17.  41.  18.   2.]
[118.  33.   5.  42.  19.   2.]
[119.  17.   1.  26.  11.   2.]
[120.  26.  11.  33.  19.   2.]
[121.  13.   7.  25.  16.   2.]
[122.  33.   7.  41.  16.   2.]
[123.  20.   6.  25.  14.   2.]
```

```
[124.  24.  12.  33.  17.   2.]
[125.  29.  11.  36.  14.   2.]
[126.  19.   7.  24.  14.   2.]
[127.  18.   9.  25.  14.   2.]
[128.  21.   7.  32.  17.   2.]
[129.  29.   9.  34.  12.   2.]
[130.  31.   7.  37.  15.   2.]
[131.  34.  17.  39.  16.   2.]
[132.  21.   7.  32.  18.   2.]
[133.  20.   7.  27.  11.   2.]
[134.  18.   5.  32.  10.   2.]
[135.  33.   9.  37.  19.   2.]
[136.  20.  13.  32.  20.   2.]
[137.  21.  10.  31.  14.   2.]
[138.  17.   9.  24.  14.   2.]
[139.  26.  10.  30.  17.   2.]
[140.  24.  10.  32.  20.   2.]
[141.  26.  10.  27.  19.   2.]
[142.  15.   6.  27.  15.   2.]
[143.  25.  11.  35.  19.   2.]
[144.  24.  12.  33.  21.   2.]
[145.  24.   9.  28.  19.   2.]
[146.  20.   4.  26.  15.   2.]
[147.  22.   9.  28.  16.   2.]

[148.  19.  13.  30.  19.   2.]
[149.  16.   9.  27.  14.   2.]]
```

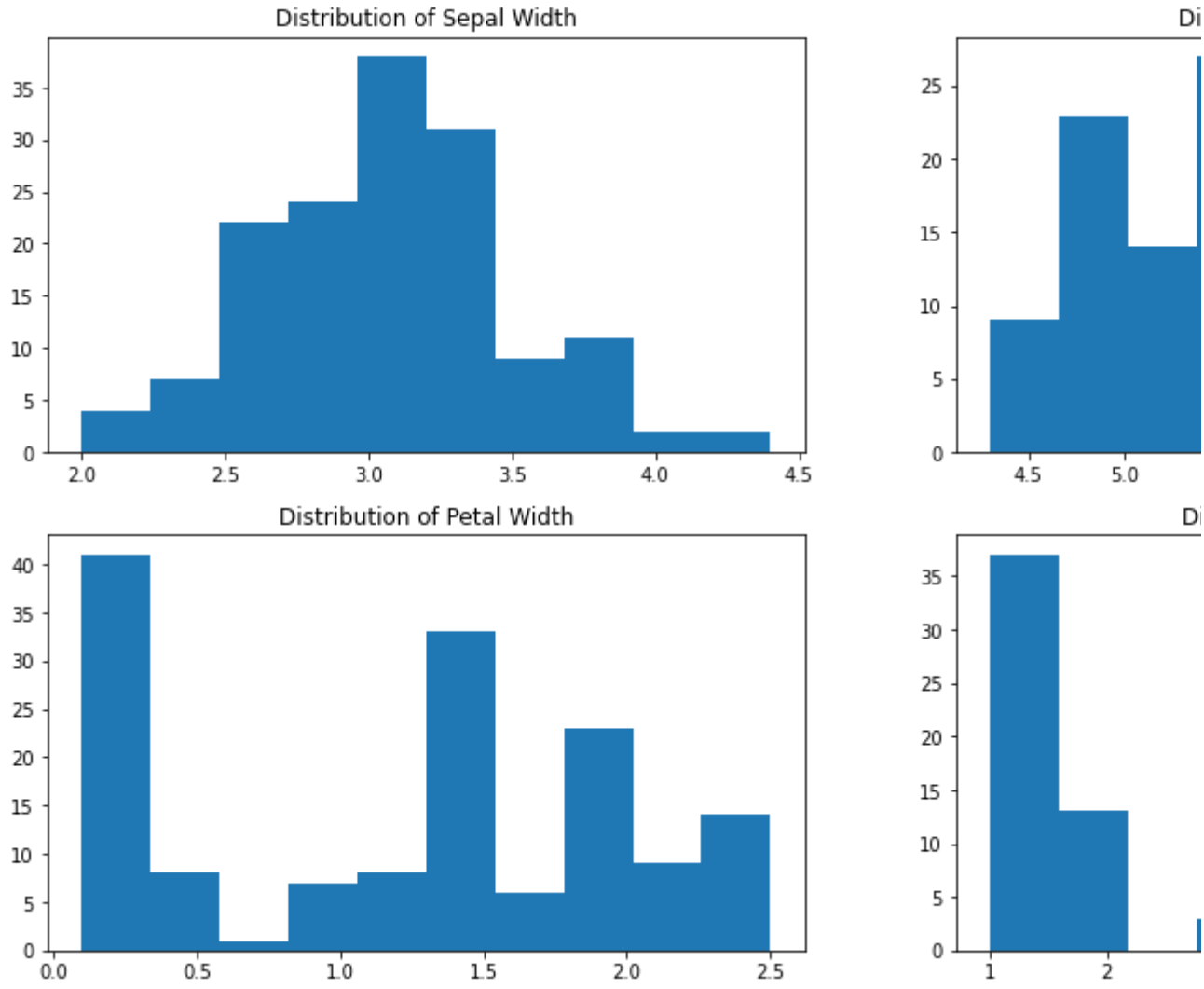
Graphs:

1. Count of Species

```
# Graph for Species Count
sns.countplot(x='Species', data=Dataset_df)
plt.show()
```

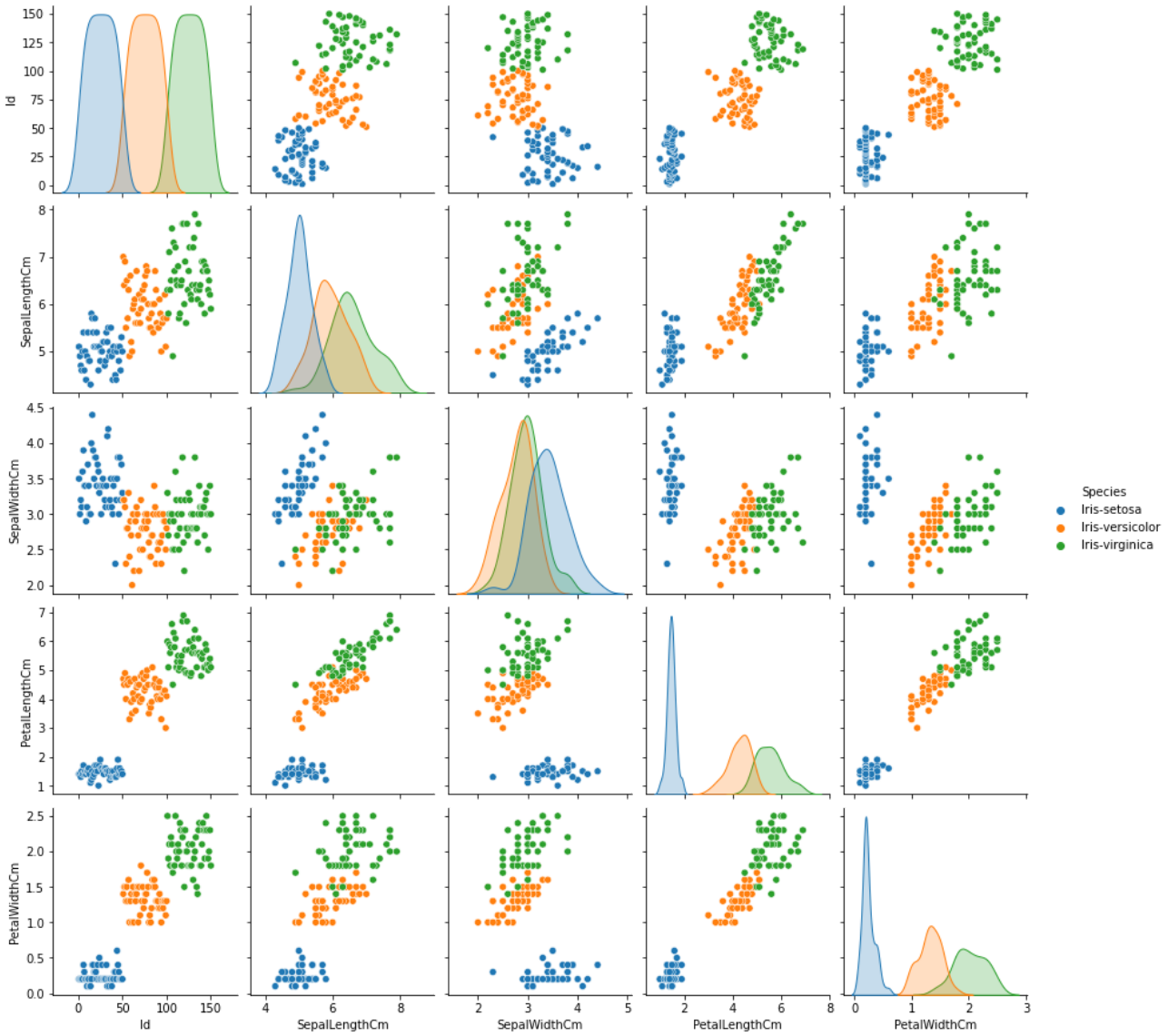
2. Histogram for Sepal Length & Width, Petal Length & Width

```
fig, axes = plt.subplots(2, 2, figsize=(16, 9))
axes[0, 0].set_title("Distribution of Sepal Width")
axes[0, 0].hist(Dataset_df['SepalWidthCm'], bins=10)
axes[0, 1].set_title("Distribution of Sepal Length")
axes[0, 1].hist(Dataset_df['SepalLengthCm'], bins=10)
axes[1, 0].set_title("Distribution of Petal Width")
axes[1, 0].hist(Dataset_df['PetalWidthCm'], bins=10)
axes[1, 1].set_title("Distribution of Petal Length")
axes[1, 1].hist(Dataset_df['PetalLengthCm'], bins=10)
plt.show()
```



3. Graph displaying a multivariate set of observations such as species and sepal and petal lengths and widths

```
sns.pairplot(Dataset_df, hue='Species')
plt.show()
```



Conclusion:

Hence, we performed operations on the given dataset, and executed data type classification, contingency, and statistical functions.

Colab paid products - Cancel contracts here

✓ 13s completed at 10:15 PM