Suraj Sawant TEB-38

Practical A10

Download the Iris flower dataset or any other dataset into a DataFrame Scan the Download the Iris flower dataset or any other dataset into a DataFrame. (e.g., https://archive.ics.uci.edu/ml/datasets/Iris). Scan the dataset and give the inference as:

1. List down the features and their types (e.g., numeric, nominal) available in the dataset. 2. Create a histogram for each feature in the dataset to illustrate the feature distributions.
2. Create a box plot for each feature in the dataset.
3. Compare distributions and identify outliers.

In [1]:
```
import pandas as pd
```

In [4]:
```
df = pd.read_csv('C:\\Users\\Admin\\Desktop\\New folder (2)\\Iris.csv')#Load the cs
```

In [21]:
```
df
```

Out[21]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa
...	...	...	...	...	...	...
145	146	6.7	3.0	5.2	2.3	Iris-virginica
146	147	6.3	2.5	5.0	1.9	Iris-virginica
147	148	6.5	3.0	5.2	2.0	Iris-virginica
148	149	6.2	3.4	5.4	2.3	Iris-virginica
149	150	5.9	3.0	5.1	1.8	Iris-virginica

150 rows × 6 columns

In [6]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149 Data
columns (total 6 columns):
 #   Column         Non-Null Count  Dtype  -
--  ------         --------------  -----   0
```

```
Id                 150  non-null     int64    1
SepalLengthCm  150 non-null    float64
2   SepalWidthCm   150 non-null    float64
3   PetalLengthCm  150 non-null    float64
4   PetalWidthCm   150 non-null    float64
5   Species        150 non-null    object
    dtypes: float64(4), int64(1), object(1)
    memory usage: 7.2+ KB
```

In [7]:
```
# Draw Histograms for each feature in dataset to illustrate the feature distributio
import matplotlib.pyplot as plt
```

In [8]:
```
plt.hist(df['SepalLengthCm'], bins=20) # Adjust the number of bins as needed here t
plt.title(f'Histogram of SepalLengthCm')# Sets the title plt.xlabel('SepalLengthCm')
# Labels the X-axis plt.ylabel('Frequency')# Labels the Y-axis
```

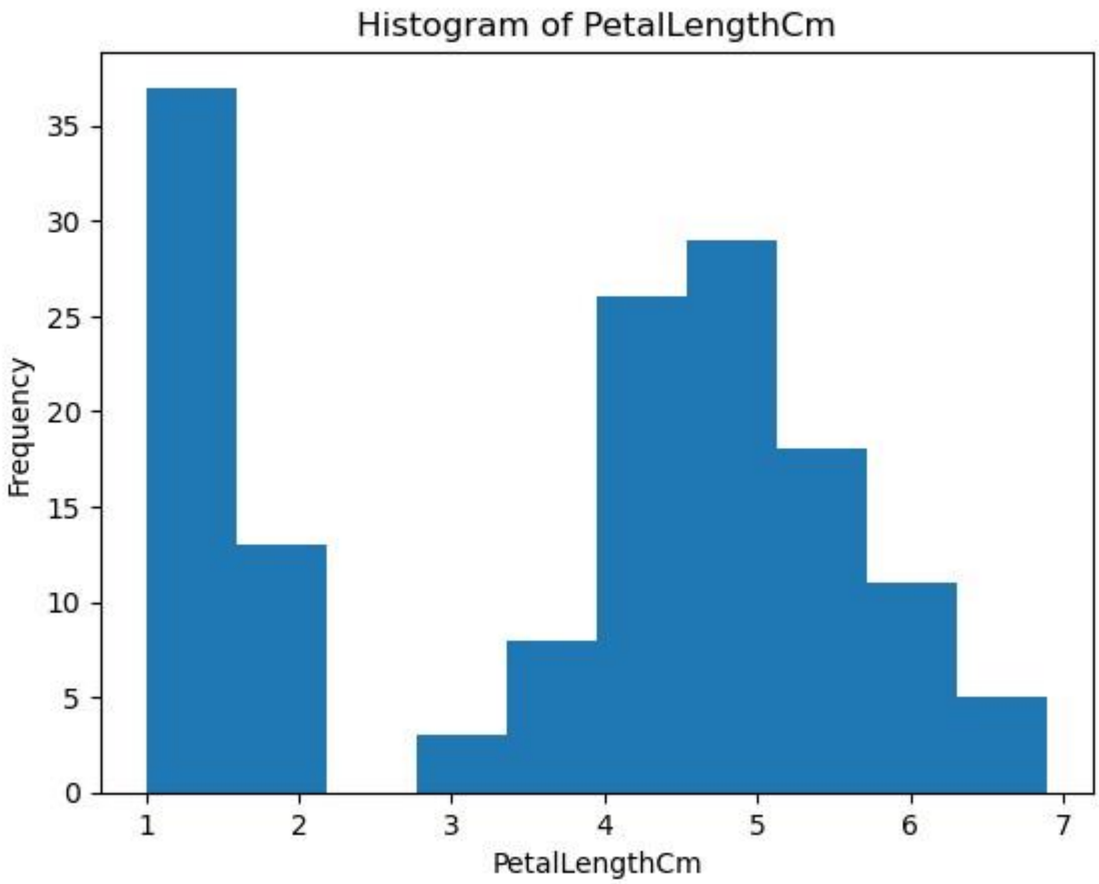
Out[8]:
```
Text(0, 0.5, 'Frequency')
```

In [9]:
```
plt.hist(df['SepalWidthCm'], bins=10) # Adjust the number of bins as needed here th
plt.title(f'Histogram of SepalWidthCm')# Sets the title plt.xlabel('SepalWidthCm')#
Labels the X-axis plt.ylabel('Frequency') # Labels the Y-axis
```

Out[9]:
```
Text(0, 0.5, 'Frequency')
```

In [10]:
```
plt.hist(df['PetalLengthCm'], bins=10) # Adjust the number of bins as needed
plt.title(f'Histogram of PetalLengthCm') plt.xlabel('PetalLengthCm')
plt.ylabel('Frequency')
```

T
ext(0, 0.5, 'Frequency') Out[10]:



In [11]:
```
plt.hist(df['PetalWidthCm'], bins=10) # Adjust the number of bins as needed
plt.title(f'Histogram of PetalWidthCm') plt.xlabel('PetalWidthCm')
plt.ylabel('Frequency')
```

T

ext(0, 0.5, 'Frequency') Out[11]:

In [12]:
```
# Create boxplots for SepalLengthCm, SepalWidthCm, PetalLengthCm, PetalWidthCm
plt.figure(figsize=(10, 6)) # Adjust figure size as needed
df.boxplot(column=['SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm', 'PetalWidthCm'
plt.title('Boxplots of Iris Features')# Adds a title plt.ylabel('Cm')# Labels the Y-
axis (unit: cm) plt.show()# Displays the plot
```

In [13]:
```
# Detect outlier in Boxplot of SepalWidthCm Column #
The boxplot itself visually represents outliers.
# We can use IQR to programmatically find them.

# Calculate Q1, Q3, and IQR
Q1 = df['SepalWidthCm'].quantile(0.25)
Q3 = df['SepalWidthCm'].quantile(0.75) IQR
= Q3 - Q1
```

In [14]:
```
# Define bounds for outliers lower_bound
 = Q1 - 1.5 * IQR upper_bound = Q3 + 1.5
 * IQR
```

In [15]:
```
lower_bound
```

Out[15]: 2.05

In [16]:
```
upper_bound
```

Out[16]: 4.05

In [17]:
```
# Identify outliers outliers = df[(df['SepalWidthCm'] < lower_bound) |
 (df['SepalWidthCm'] > upper_boun
```

In [19]:
```
# Print or further process the outliers
 print("Outliers in SepalWidthCm:") outliers
```

Outliers in SepalWidthCm:

Out[19]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
15	16	5.7	4.4	1.5	0.4	Iris-setosa
32	33	5.2	4.1	1.5	0.1	Iris-setosa
33	34	5.5	4.2	1.4	0.2	Iris-setosa
60	61	5.0	2.0	3.5	1.0	Iris-versicolor

In [20]:
```
# Printing the Row Indexes of Outliers
 outlier_indices = outliers.index
```

```
print("Outliers RowIndex:")
outlier_indices
```

```
Outliers RowIndex:
Int64Index([15, 32, 33, 60], dtype='int64')
```

Out[20]:

With above commands, the practical problem statements are fnished. But to study the the Iris Flower in the perspective of Biologists, the morphological (structural) differences between the s
p

```
# Lets do a comparitive analysis of all species on PetalWidthCm\
# Draw Specieswise Boxplot for PetalWidthCm import
seaborn as sns sns.boxplot(x='Species',
y='PetalWidthCm', data=df) plt.title('Species-wise
Boxplot of PetalWidthCm') plt.show()
```

are to be studied. Below is the extended code (not for exam)

In [22]:

In [32]:

```
# Create a histogram of PetalWidthCm for each species for
species in df['Species'].unique():
    species_data = df[df['Species'] == species]
plt.hist(species_data['PetalWidthCm'], bins=10, alpha=0.6, label=species, edgec
```

```
plt.title('Histogram of PetalWidthCm by Species') plt.xlabel('Petal
Width (cm)') plt.ylabel('Frequency') plt.legend(title="Species")  #
plt.grid(axis='y', linestyle='--', alpha=0.6) plt.show()
```

In [34]:
```
# Calculate IQR and identify outliers for each species for PetalWidthCm def
find_outliers_iqr(data):
    Q1 = data.quantile(0.25)
    Q3 = data.quantile(0.75)
IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR
    outliers = data[(data < lower_bound) | (data > upper_bound)]
return outliers
```

In [35]:
```
lower_bound
```

Out[35]: 2.05

In [36]:
```
upper_bound
```

Out[36]: 4.05

```
In [38]: print("Printing Outlier for each species for PetalWidthCm")
          for species in df['Species'].unique():
    species_data = df[df['Species'] == species]['PetalWidthCm']
outliers = find_outliers_iqr(species_data)     print(f"Outliers for
{species}: {outliers.values}") Printing Outlier for each species for
PetalWidthCm
```

```
Outliers for Iris-setosa: [0.5 0.6] Outliers
for Iris-versicolor: []
O
```

utliers

for

Iris-

```
# Return row indices for outliers for Iris-setosa def
find_outlier_indices(df, species_name, column_name):
    species_data = df[df['Species'] == species_name][column_name]
outliers = find_outliers_iqr(species_data)     outlier_indices =
outliers.index     return outlier_indices
```

virginica: [] In [40]:

```
In [41]: setosa_outlier_indices = find_outlier_indices(df, 'Iris-setosa', 'PetalWidthCm')
          print(f"\nRow indices of outliers for Iris-setosa in 'PetalWidthCm': {setosa_outlie
```

R

ow

indices of outliers for Iris-setosa in 'PetalWidthCm': [23, 43] In [ ]: